WORDT
NIET UITGELEEND

Output Prediction for non Demand-Driven Power Systems using Neural Networks

B.I.J. Siljee

Rijksuniversiteit Groningen
Bibliotheek Wiskunde & Informatica
Postbus 800
9700 AV Groningen
Tel. 050 - 363 40 01

Master's thesis
Supervisors: Drs. ing. J.K. Kok and Dr. ir. J.A.G. Nijhuis
Department of Mathematics and Computing Science
University of Groningen
May 2003

Rijksuniversiteit Groningen
Bibliotheek Wiskunde & Informatica
Postbus 800
9700 AV Groningen
Tel. 050 - 363 40 01

Abstract

In the Dutch liberalized electricity market, parties involved in the use of the electricity network need to send their planned electricity activities to the national grid administrator each day in advance. When a market party creates imbalance by deviating from the planned activity, it has to pay imbalance costs. The output of non demand-driven power systems is irregular and depends on different stochastic factors. To minimize the producer's imbalance costs, the power output has to be accurately predicted.

This thesis investigates the output prediction of non demand-driven power systems using neural networks. Models based on the multilayer perceptron neural network are used for this purpose. The results obtained when building models based on real-life data, as well as the simulation of the models in use are described. A detailed analysis of the influence of various parameters in neural network based modeling is given, like input relevance and the optimal division of one year for separate models.
It is shown that ensembles of these neural networks are able to generalize from real-life data and achieve a reasonably well performance in predicting the power output.

WORDT
NIET UITGELEEND

Samenvatting

In de geliberaliseerde elektriciteitsmarkt van Nederland moeten marktpartijen die gebruik maken van het elektriciteitsnet elke dag hun geplande elektriciteitsactiviteiten naar de landelijke netbeheerder sturen. Als een marktpartij onbalans creëert door af te wijken van de geplande activiteit, dan moet zij onbalanskosten betalen. De productie van niet-vraaggestuurde energiesystemen is onregelmatig en hangt af van verschillende stochastische factoren. Om de onbalanskosten van de producent te minimaliseren moet de elektriciteitsproductie nauwkeurig voorspeld worden.

Deze scriptie behandelt de voorspelling van de productie van niet-vraaggestuurde energiesystemen met behulp van neurale netwerken. Voor dit doel worden modellen op basis van multilayer perceptron neurale netwerken gebruikt. De resultaten die verkregen zijn bij het maken van de modellen op praktijkdata en tijdens de simulatie van de toepassing van de worden beschreven. Bovendien wordt er een gedetailleerde analyse gegeven van de invloed van de verschillende parameters van het modelleren met neurale netwerken, zoals input relevantie en de optimale jaaropdeling voor aparte modellen.
Er wordt gedemonstreerd dat ensembles van deze neurale netwerken praktijkdata goed kunnen generaliseren en bovendien een redelijk goed resultaat behalen in het voorspellen van de elektriciteitsproductie.

Acknowledgements

This thesis is the result of research conducted at the Energy research Centre of the Netherlands (ECN). I am grateful to Koen Kok, my supervisor at ECN, for his availability, support and guidance, and the many useful discussions.

I also wish to thank my supervisor Jos Nijhuis at the Department of Computer Science, University of Groningen, for his comments and ideas.

Special thanks go to Berto Booijink, for the almost weekly Friday night discussions and for his thoroughness in hunting down English imperfections in this thesis.

Furthermore, I would like to thank my parents, my sister, and my friends for their support and the many trips to come and visit me during my stay in the far West.

CONTENTS

Chapter 1 Introduction

Electricity plays a very important and ever growing role in modern society. Already the scale of electricity production and demand has grown to such a degree that its impact on the environment has become noticeable. The emission of CO_2, NO_x, and SO_x into the air, water pollution and solid waste from nuclear and coal-fired plants are only a selection of worrisome effects. Furthermore, the expectation towards fossil fuels used nowadays is a decrease in supply or at least a huge rise in costs to obtain them.
Technological improvements provide alternatives to fossil fuels in ways of more sustainable (or renewable) electricity production systems, like more effective power plants and non demand-driven power systems. Examples of the latter are solar cells (photovoltaic), wind turbines, and combined heat and power systems. The electricity output of these systems is fluctuating and caused by stochastic variables.

With the liberalization of the electricity market the problem arises of how to organize an electricity trade market guaranteeing the maneuverability of the power system in respect of system balancing. TenneT, the Dutch national network operator, corrects any imbalances or internal transmission constraints generated in the system. To be able to do this, the "energy-programme responsibility" was set up. The program responsibility basically is a system in which all market parties involved in the use of the network send their planned production, consumption, and transportation of electricity to TenneT on a daily basis.

When a producer does not produce according to the planning he registered the previous day, he has to pay compensations for imbalance (the difference between expected and actual power output) to TenneT. Predictions made for non demand-driven power systems are hardly ever equal to the actual power output, which leads to large imbalance costs. Therefore it is important to use models that can give an accurate prediction of the power output of non demand-driven power systems.

In this thesis we investigate the modeling and predictability of this type of prediction problems.

1.1 Problem Description

When electricity is fed into the national power system, it has to be registered in advance to TenneT. But the amount of electricity output of non demand-driven power systems depends on different stochastic factors, like certain weather characteristics. Power production by photovoltaic (PV) cells clearly depends on sun radiation, and the production of wind turbines on wind speed, while both may also depend on other elements like temperature and humidity. Combined Heat and Power (CHP) or cogeneration is the simultaneous production of heat and electricity, and applications of CHP plants are often in the field of horticulture, as well as hospitals, industries, and local district heating systems for cities. When users turn on their CHP installations, i.e. when they produce heat and power, depends on weather and time factors.

The production of these non demand-driven power systems is highly influenced by weather conditions. The models we want to develop for the prediction of the power production will therefore have to be based on time-series of weather forecasts.

A power prediction is needed for the following two reasons:

- The electricity production plan required by TenneT to maintain system balance is to be submitted at noon for the entire following day, from midnight to midnight. In practice the prognosis is drawn up several hours before noon, using production and weather measurements until midnight the night before, see Figure 1-1. This leads to a prediction interval from 24 to 48 hours in advance. The submitted production plan has to be specified for every quarter of an hour within the interval.
- Producers of electricity have to find a buyer themselves. The sooner is known how much will be produced; the sooner the search for a buyer can start.

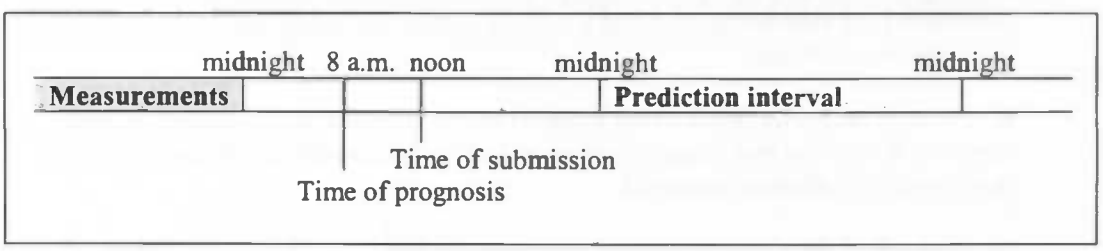

Figure 1-1 Prediction time scale

The kind of power systems referred to are aggregations of many (smaller) installations, and these installations can be grouped together or distributed all over the country. Only the power output of all installations together needs to be predicted. But because of the number of installations and their stochastic behavior, the exact underlying physical relation between the weather forecasts and the power output of the whole park is unknown. A lot of modeling techniques require this knowledge in order to set up mathematical equations. We will not be able to use these techniques; we need a model that can infer functional relationships from only data by itself (also called data-driven modeling). Neural networks do not need functions to be fixed in order to learn, so we will use those as the modeling tool to build the predictor.

1.2 Goal

We want to create a modeling technique for models that accurately predict the output of non demand-driven power systems, in order to minimize the imbalance costs of the prediction. For this purpose, we investigate how to use neural networks to model and predict time-series formed by weather forecasts and the resulting power output.

Objectives

This thesis addresses the following questions:

1) Which neural networks are suited for the modeling and prediction of these time-series?
2) How can statistical characterization and data analysis methods be used for preprocessing and feature extraction?
3) What measurements quantify the performance of the neural networks?
4) Prediction of power output. How well can the neural networks predict future output from output histories and weather forecasts, and what are their limitations?

1.3 Methodology

We will use neural networks as the modeling tool to build the predictor. As possible realizations of neural networks the multilayer perceptron, time-delay neural networks, and ensembles of neural networks are investigated. We implement and test various input variable selection techniques to improve the performance of the neural networks.

We achieve the second objective by studying the weather and power output time-series. By applying techniques from summarization, sequence analysis, and dependency modeling we characterize the time-series, and find a way for feature extraction and data de-correlation as preprocessing methods.

We evaluate the performance of the models using a statistical technique and an economical measure. We use the root mean square error and the imbalance cost of the error of the predictions as evaluation measures.

We apply all of the above-mentioned techniques to data from a real-life problem of power prediction, the modeling data. We evaluate the results of these experiments by simulating the use of the models in reality with a second data set, the simulation data, obtained from the same problem. We discuss the performance and generalization ability of the models.

1.4 Accomplishments

We used the Fourier transform and autocorrelation to extract those features from the data that contain the time factors. By doing so we created a static regression problem instead of the dynamic time-series. Multilayer perceptrons were then used to predict the power output. We discovered that although the de-correlation of the input data by applying the Principal Component Analysis yields a better error performance of 7 % on the modeling data from the real-life problem, it causes a performance decrease on the simulation data. We implemented single neural networks and ensembles of neural networks, and the latter resulted in 32% less imbalance costs on the simulation data. It turned out that a linear input relevance algorithm selects the input variables based on which the models obtain the best results. The best

generalizing models were the ones trained and tested on the complete modeling data. The imbalance costs were calculated on the simulation data, and the best model has a total of €6068 of imbalance costs, summed over the six available months. This is a 10% decrease with respect to the imbalance costs of the runner-up. This model also has a good generalization performance, as the prediction error hardly increases on the simulation data in comparison with the error on modeling data.
Comparing the model to a persistence model that predicts the power output as the measured power output of two days ago, we achieved a performance increase of 49%.

1.5 Overview

In chapter 2, an introduction to neural networks is given. We present two different kinds of neural networks used for prediction: multilayer perceptrons and time-delay neural networks. Next we discuss techniques for the determination of the most relevant input variables and methods to create ensembles of neural networks. In chapter 3, we describe four methods often used for data analysis: summarization, regression, sequence analysis, and dependency modeling. These techniques are applied to the power prediction of non demand-driven power systems in general, and to data from a real-life problem. We first discuss the statistical characteristics of the raw data, and continue with the sequence analysis of the power output history and dependency modeling of the weather components. Chapter 4 starts with a discussion of various methods for measuring the performance of neural networks when applied to power prediction. This is followed by the application of the work from chapters 2 and 3 to the problem of the power prediction of the real-life problem. First we discuss the input relevance techniques. After that we show the results of modeling and predicting the time-series with the multilayer perceptron and ensembles of multilayer perceptrons. We conclude with a discussion of the results obtained. In chapter 5 we simulate the on-line use of the models resulting from chapter 4, and investigate their generalization performance. In the last chapter we summarize the results from the previous chapters and draw conclusions. We also discuss various interesting possibilities for further research.

Chapter 2 Neural Networks

Artificial neural networks have been studied since the work of McCulloch and Pits in 1943 [16]. Learning algorithms for neural networks that can be used for time-series modeling and prediction were not developed until the 1980's [19].
This chapter starts with a general description of neural networks. After that, a selection of neural networks for time-series prediction is given. This is followed by a discussion of input relevance determination techniques. Ensembles of neural networks are covered in the last section.

2.1 Introduction to Neural Networks

Artificial neural networks (ANNs) are based upon the working of the real neural networks: the nervous systems of humans and animals. The neurons in an ANN are simple units that receive an input signal from other neurons and create a single output signal. The creation of the output signal is based on information stored internally or arriving via weighted connections between the neurons. The resulting output signal can be input to other neurons. Figure 2-1 is a schematic picture of an artificial neuron.

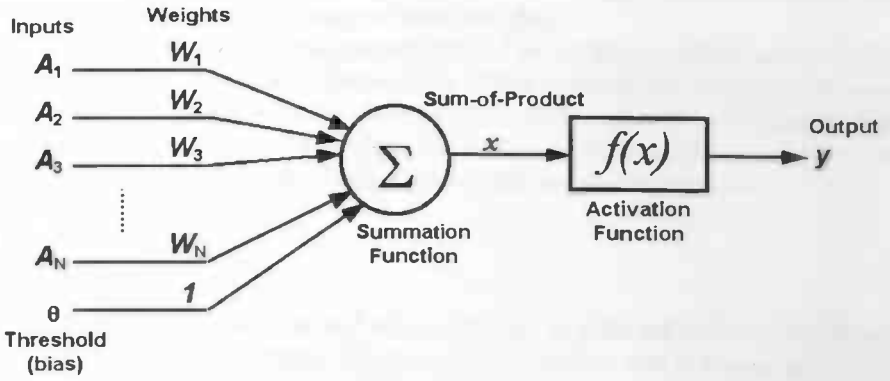

Figure 2-1 Artificial Neuron

The summation function associates each input with its weight, of which the value is between 0 and 1. The summation often has an extra input value θ associated with a weight value of 1 to represent the threshold or *bias* of a neuron.

The summation function is

$$x = \sum_{i=1}^{N} A_i W_i + \theta$$

where N is the number of connections, A_i the input value on connection i, W_i the weight of connection i, θ the bias, and x the weighted sum of the input connections of the neuron.

Network architecture

ANNs obtain their power from the combination of many neurons. The way neurons are combined into a network is called network architecture. Haykin [4] divides them into three different classes:

1. **Single-Layer Feedforward Networks**
   In a layered neural network the organization of neurons is in the form of layers. The simplest form has an input layer of source nodes that projects onto an output layer of computation nodes. This is a strictly feedforward network in the sense that no information goes backwards. Since there is only one layer with computation nodes, this is called a single-layer network.
2. **Multilayer Feedforward Networks**
   This class of neural networks has one or more hidden layers with hidden neurons. A neural network can extract higher order statistics when one or more hidden layers are added. The input layer nodes supply the input signals for the second, hidden layer. The output signals of the second layer go to the third. This goes on until the output layer is reached.
3. **Recurrent Networks**
   A recurrent network has at least one feedback loop, in contrary to the feedforward networks, which have none. The output signal of a neuron can be fed back to the same neuron or to a neuron in a previous layer.

Learning

The property that makes neural networks so interesting is their ability to learn from their environment, and to improve their performance through learning. A neural network learns through an interactive process of adjustments applied to its synaptic weights and bias levels. During this process the network is repeatedly provided with a set of input patterns together with the corresponding output patterns. This set of input and output patterns is called a training set. At each step of the training process the weights and bias levels of the network may be changed according to a learning algorithm. There are several learning algorithms; most of them are based on error-correction of the output.

A network is said to *generalize* well when the input-output mapping computed by the network is (almost) correct for test data never used in creating or training the network. When a network is *overfitting* (i.e. memorizing the training data exactly), it loses its ability to generalize between similar input-output patterns. Networks may be overfitting by finding a feature (for example due to noise) that is present in the training data but is not true to the underlying function that is to be modeled. This phenomenon is a common problem with neural networks and machine learning algorithms in general. To define a measure for the generalization ability is difficult. There are some theoretical approaches on this issue in [4] and [8].

Furthermore, properly trained neural networks have a good interpolation performance, but a very poor extrapolation performance. Unfortunately there is no general rule to find out if a new pattern is within the interpolation space [5].

There are a lot of different neural network types used for prediction. To give an exhaustive overview of all would fall outside the scope of this thesis, for that we refer to [4]. The networks covered in the next section are a selection.

2.2 Multilayer Perceptrons

Model

The multilayer perceptron (MLP) is a very important class of neural networks. Typically, the network consists of a set of source nodes that constitute the input layer, one or more hidden layers of computation nodes, and an output layer of computation nodes. The input signal propagates through the network in a forward direction, on a layer-by-layer basis. Multilayer perceptrons are trained in a supervised manner and often with an algorithm known as the error back-propagation algorithm. During the training process, the weights are adapted backwards. An MLP has three distinctive characteristics [4]:

1. The model of each neuron in the network includes a nonlinear, smooth activation function. A commonly used form of nonlinearity is the S-shaped sigmoid activation function:

$$Y_j = \frac{1}{1+e^{-x_j}}$$

   where x_j is the induced local field (i.e. the weighted sum of all synaptic inputs plus the bias) of neuron j, and Y_j is the output of the neuron.
2. The network contains one or more layers of hidden neurons that are not part of the input or output of the network. These hidden neurons enable the networks to learn complex tasks by extracting progressively more meaningful features from the input patterns.
3. The network shows a high degree of connectivity, determined by the synapses of the network. A change in the connectivity of the network requires a change in the population of synaptic connections or a change of their weights.

The combination of nonlinearity and connectivity of the network and the use of hidden neurons make it hard to understand the behavior of the network. But it is the combination of these characteristics, together with the ability to learn from experience by means of training that account for the computing power of the MLPs. They have been shown to be a "universal approximator" [9]: theoretically they can approximate any desired function.

Prediction

MLPs are well suited for prediction and widely applied in different kinds of predictions [12] [14] [17]. This is because advantages like their high flexibility and independence of a priori assumptions when estimating a noisy nonlinear function.

2.3 Time-Delay Neural Networks

Model

Time-Delay Neural Networks (TDNNs) use a technique to supply neural networks with "memory", as a way to deal with the temporal dimension. This is done by the introduction of

time delays on connections. In other words, through delay, inputs arrive at hidden units at different points in time, thus being "stored" long enough to influence subsequent inputs. The primary role of memory is to transform a static network into a dynamic one. In particular, by embedding memory into the structure of a static network such as an ordinary MLP, the output of the network becomes a function of time. This way the static network accounts for nonlinearity, and the memory accounts for time.
In the context of neural networks, there are two types for temporal processing [4]:

1. **Time lagged feedforward networks (TLFN)** are basically multi-layered feedforward networks like the MLP, with time delays added to account for the temporal processing.
2. **Recurrent neural networks (RNN)** also have a topology similar to MLPs, but these networks have one or more feedback loops. The use of feedback has the potential of reducing the memory requirements significantly, as compared to static networks.

Prediction

These kinds of networks are especially useful for the prediction and modeling of time-series. TDNNs have the computational power of MLPs and can extract the periodic features from data. As a result less preprocessing will be necessary at the cost of a larger network. Preprocessing requires prior knowledge of data properties, while a TDNN can create the delays in the training process.

2.4 Universal Myopic Mapping Theorem

In [21] Sandberg and Xu gave the mathematical justification for the use of both the MLP and the TLFN for time-series prediction. They showed that under certain conditions any shift-invariant myopic map can be uniformly approximated arbitrarily well by a structure of the form depicted in Figure 2-2. This structure is a universal mapper. The blocks labeled $\{h_j\}_{j=1}^{L}$ represents a bank of linear filters operating in parallel. The h_j are drawn from a large set of real-valued kernels, of which each one represents the impulse response of a linear filter. The block labeled $\mathcal{N}$represents a static (memoryless) nonlinear feedforward network, like an ordinary MLP.
One of the conditions is that the map is myopic: it has a uniform fading memory. It is further assumed that the map is causal. For a map to be causal, the output signal at step $n = 0$ may not depend on input signals applied at $n > 0$. For a system to be shift-invariant the following must hold: if the output $y(n)$ is obtained as a result of the input $x(n)$, then the delayed input $x(n\text{-}n_0)$ must have output $y(n\text{-}n_0)$, where the time shift n_0 is an integer.

The universal myopic mapping theorem holds under the conditions that the map is causal, and can be formulated as follows [22]:

Theorem 2.4.1
Any shift-invariant myopic dynamic map can be uniformly approximated arbitrarily well by a structure of two functional blocks: a bank of linear filters feeding a static neural network.

Sandberg and Xu showed that for any single-variable, shift-invariant, causal, uniformly fading memory map there exists a gamma memory and static network of which the combination approximates the map uniformly well. The combination is a structure of a linear preprocessing stage followed by a nonlinear network without memory. This structure is inherently stable, provided that the linear filters are stable. Thus, the universal myopic

mapping theorem provides a separation of the roles of short-term memory and memoryless nonlinearity.

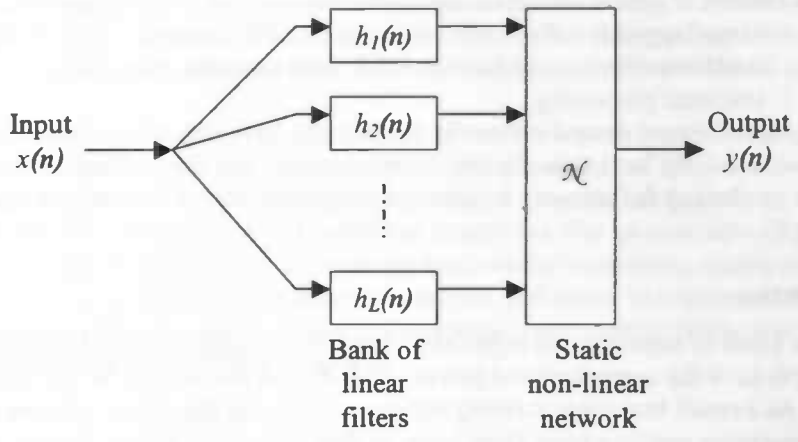

Figure 2-2 Generic structure for the universal myopic mapping theorem

Much research has already been done on MLPs; less is known about the techniques for training and optimization of TLFNs. If we preprocess the input data to extract all the features that capture the time components of the time-series, then we can use an MLP as the nonlinear network without memory. The use of the preprocessing stage converts the problem from time-series prediction to a nonlinear regression problem, which is static. Using MLPs to solve regression problems is probably easier than using TLFNs to predict time-series, while according to theorem 2.4.1 we can expect to obtain the same results.

2.5 Input Relevance Determination

A neural network with a small input dimension is less likely to learn noise in the training data, and may thus generalize better to new data. Removing the irrelevant input variables reduces the input dimension. Furthermore, relevance information increases the user's understanding of the problem. Since every input variable is either relevant or not, we have 2^N possibilities for N input variables. Training neural networks for every subset of input variables is only feasible when the number of input variables is very small. Alternatives to determine the relevant input variables for this brute force method are the approximations forward selection, stepwise selection, and backward elimination.

1. *Forward selection* starts without any input variables, and sequentially adds the most relevant input to the network, stopping when the network does not significantly improve performance according to some given standard, or when the network performance exceeds a certain threshold. Relevance determination can thus be stopped in an early stadium; when the required performance is reached there is no need to continue adding inputs.
2. *Stepwise selection* is an altered version of forward selection where at every step the variables included on the previous steps can be reconsidered, since when a new variable in included, other variables included in earlier steps may become irrelevant.

3. *Backward elimination* removes the least relevant input from a network trained on all input variables, one at the time. This process is stopped when the performance of the network drops below a given threshold by removing any of the remaining input variables. In [13] an overview of backward elimination algorithms is given.

We will consider two examples of forward selection and one example of backward elimination in the next three sections.

2.5.1 Linear Input Relevance Determination

A fast, but naïve way to determine the relevance of inputs is to create linear regression models, also referred to as stepwise regression. The most relevant input of N input variables is determined by creating N models, each consisting of one of the N inputs, and keeping the input of the model with the best performance. Next this procedure is repeated with the remaining N-1 input variables, which are combined with the most relevant input from the previous step into N-1 regression models of two variables. This process is reiterated until the improvement in performance is not significant anymore according to some standard. This is a typical forward selection algorithm.
Linear input relevance determination is very fast compared to input relevance methods based on neural networks, but its major setback is the total disregard of nonlinear relations between the input and output variables.

2.5.2 Naïve Input Relevance Determination

To include nonlinear relations in the relevance determination, a still naïve but more computationally expensive version of forward selection can be used. The most relevant inputs are determined in the same sequence of steps as described above for the linear input relevance determination, but instead of creating linear regression models, neural networks are trained for each subset. The performance of neural networks is usually very dependent on the initialization of the weights and the division of the training and test data set. Therefore, this relevance determination method should be repeated several times to see if the resulting selection of relevant inputs is reproducible, making it even more computationally expensive.

2.5.3 Input Selection with Partial Retraining

In [13] Van de Laar proposes an input relevance determination algorithm called partial retraining. This algorithm is based on the assumption that a neural network trained on all N input variables has constructed a good representation of the data in its hidden layers. The goal is then to find a new neural network based on N - 1 input variables, with hidden-layer activities as close as possible to the original one. This is achieved by adjusting the weights of the network in one retraining step. Partial retraining is far less computationally expensive than the naïve approach. This reduction is accomplished by using implicit knowledge about the problem, which is contained in the neural network trained on all input variables. We chose this algorithm because it outperformed all other backward elimination algorithms such as Optimal Brain Surgeon, weight analysis, and constant substitution in a benchmark study [13].

2.6 Ensembles of Neural Networks

When the complete data set is split up into a training and validation set, and training is stopped when the error on the validation set starts increasing, then the final network depends on the accidental subdivision in training and validation set, and also on the, usually random, initialization of the weights. Neural networks trained on different parts of the available data will thus have different, maybe even contradicting solutions. Therefore, it is advisable to apply the same procedure several times on a different part of the data starting from different initial weights. This technique is often referred to as training ensembles of neural networks.

In these ensembles of neural networks, the outputs of various neural networks that are trained to perform the same task are combined to form a combined prediction. Neural network ensembles were first introduced in [3]. Since then, a lot of research has been done in this field, both on linear [6] [27] and nonlinear [7] combined ensembles. For an overview of possible combination methods see [10]. We will use linear methods for combining neural networks. Linear methods are often simpler to understand, analyze, and easier to implement than nonlinear methods, while their performance still is a significant improvement over the average performance of the ensemble members.

2.6.1 Linear ensembles

A linear ensemble is a linear combination of the 'members' of the ensemble, as depicted in Figure 2-3. All the members are trained on bootstraps of the same data set; they may differ from each other in the choice of initial conditions used in network training. To understand how linear ensembles improve performance, we will first give a short explanation of the bias/variance dilemma.

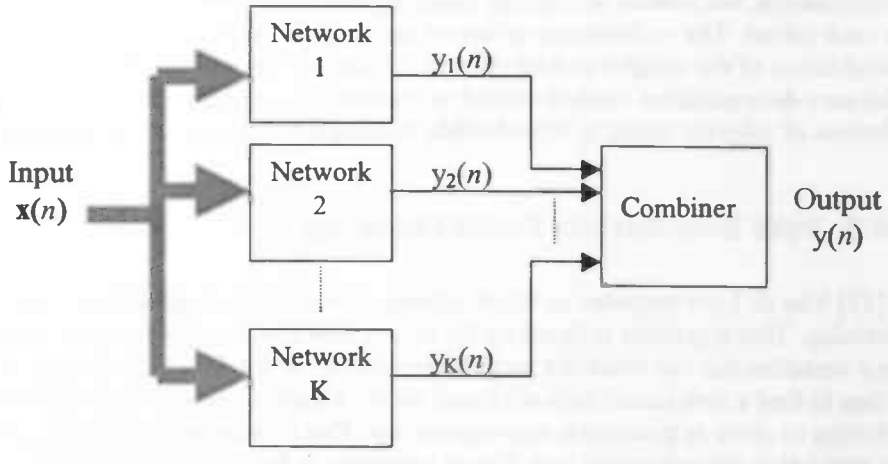

Figure 2-3 Block diagram of a linear ensemble

Bias/variance dilemma

Consider first the case of a single neural network that has been trained on a given data set. Let the vector x denote an input signal not seen before, and let *d* denote the corresponding desired

response; x and d represent realizations of the random vector **X** and random variable D, respectively.
The mean square error between the expectation $E[D \mid \mathbf{X} = \mathbf{x}]$ and the input-output function $F(\mathbf{x})$ realized by the neural network, can be expressed as [4]:

$$E_S[(F(\mathbf{x}) - E[D \mid \mathbf{X} = \mathbf{x}])^2] = B_S(F(\mathbf{x})) + V_S(F(\mathbf{x}))$$

The expectation E_S is taken over the space S, defined as the space encompassing the distribution of all training sets (i.e. inputs and target outputs) and the distributions of all initial conditions.
The term $B_S(F(\mathbf{x}))$ is the *bias* squared:

$$B_S(F(\mathbf{x})) = (E_S[F(\mathbf{x})] - E[D \mid \mathbf{X} = \mathbf{x}])^2$$

and expresses the inability of the neural network defined by $F(\mathbf{x})$ to accurately approximate the desired response. The term $V_S(F(\mathbf{x}))$ is the *variance* of the approximation function $F(\mathbf{x})$:

$$V_S(F(\mathbf{x})) = E_S[(F(\mathbf{x}) - E_S[F(\mathbf{x})])^2]$$

To achieve good overall performance, the bias and the variance would both have to be small. Unfortunately, for single neural networks that learn from a training set of fixed size, the price for achieving a small bias is a large variance, and vice versa. This is the bias/variance dilemma.
Let now $F_I(\mathbf{x})$ denote the average of the input-output functions of the networks in Figure 2-3 over a "representative" number of initial conditions, and S' the distributions of all training sets. In [4] it is shown that

$$B_{S'}(F_I(\mathbf{x})) = B_S(F(\mathbf{x})) \tag{2.1}$$

and

$$V_{S'}(F_I(\mathbf{x})) \leq V_S(F(\mathbf{x})) \tag{2.2}$$

Thus, from equations (2.1) and (2.2) we draw two conclusions [18]:
1. The bias of the members is reduced at the cost of variance by purposely overfitting the individual members of the ensemble.
2. The variance is reduced by using different initial conditions in the training of the individual members, and then linearly combining their outputs by averaging. The bias remains equal to the bias of the single neural networks.

Linear ensembles improve error performance by the combined use of these two effects.

To optimize ensemble error performance, it is important that good values for the combination factors of the ensemble members are selected. Two often-used methods are:
- Bagging [2], where the prediction on a newly arriving input vector is the average over all network performances. The performance of the individual networks on the data set used for training and stopping is completely disregarded.
- Bumping [25], where all networks are thrown away except the one with the lowest error on the complete data set.

Balancing [6] is an intermediate form of bagging and bumping. Each network receives a weighting factor α_i, which depends on the expected performance of the network on a new set

of p_{test} test patterns. The prediction of all networks on pattern ν is defined as the weighted average

$$\tilde{m}^{\nu} \equiv \sum_{i=1}^{n_{run}} \alpha_i \tilde{o}_i^{\nu}$$

where $\tilde{o}_i$ is the network output for each network i on p_{test}.

The goal is to find the weighting factors α_i, subject to the constraints

$$\sum_{i=1}^{n_{run}} \alpha_i = 1 \text{ and } \alpha_i \geq 0 \quad \forall i ,$$

yielding the smallest possible generalization error

$$E_{test} \equiv \frac{1}{p_{test}} \sum_{\nu=1}^{p_{test}} \left(\tilde{m}^{\nu} - \tilde{t}^{\nu} \right)^2 .$$

The estimates for the generalization errors are based on the network performances on validation data, see [6].

Chapter 3 Data Analysis

There are several methods for data interpretation and modeling. In [26] an overview of these methods, like summarization, regression, sequence analysis, and dependency modeling, is given. We will first give a description of these methods, and next discuss their application to the prediction of the output of non demand-driven power systems. In the last part of this chapter we will use the analysis methods to preprocess real-life data from an output prediction of non demand-driven power systems problem.

3.1 Summarization

Summarization provides a compact description for a subset of data. Simple techniques include mean and standard deviations; more sophisticated are summary rules, multivariate visualization techniques, and functional relationships between variables.

3.2 Regression

Linear and nonlinear regression methods are among the most common approaches for correlating data. Statistical regression methods, like autoregressive moving average (ARMA) models and Nonlinear Least Squares, often require the user to specify a function over which data is to be fitted. In order to specify the function, it is necessary to know the forms of equations governing the correlation for the data. However, if prior knowledge is not available, it is necessary to find out the most probable function by trial-and-error, which may require very time consuming effort. Neural networks do not need functions to be fixed in order to learn and have shown very remarkable results in representing nonlinear functions [4] [20] [23] [26], as long as there is enough training data available. But the function resulting from using neural networks is not easy to understand, if there is a function at all. Most of the time the system virtually is a black box without any explanations.

3.3 Sequence analysis

Sequence analysis techniques, also called time-series data techniques, are first of all useful for very short range forecasting. But the further away in time the prediction has to be, the smaller the accuracy will be [24]. Another major application of sequence analysis techniques is the preprocessing of data. Preprocessing involves using a minimum of data points to capture the features of the data and to remove its noise. This includes techniques like filters, Fourier and wavelet transforms, statistical approaches, neural networks, as well as various qualitative signal interpretation methods. Two important techniques are Fourier transforms and autocorrelation, a short description of both is given below.

Fourier transforms

Fourier transforms are well known as a useful technique for frequency analysis of a signal, which breaks down into constituent sinusoids of different frequencies. The dominant frequencies make up a big part of the signal. By applying Fourier transforms on the data one can discover the important features (dominant frequencies) of the data and remove the irrelevant ones (noise). This way the data dimension is reduced. [26]

Autocorrelation

The autocorrelation function is, like the Fourier transforms, a useful technique for describing stationary processes. A plot of the autocorrelation shows how the correlation between any two values of the same variable changes as their separation in time changes. The autocorrelation function is a transform of the spectral density function (the latter is the Fourier cosine transform of the auto covariance function).
Autocorrelation reflects another aspect though: if successive values are positively associated, then the autocorrelation plot is positive as well. If the data alternates, then there would be a negative association between values.

3.4 Dependency modeling

Dependency modeling describes dependencies among variables. The structural level of the model specifies which variables are locally dependent; the quantitative level specifies the strengths of the dependencies using some numeric scale. Examples of tools for dependency modeling include probabilistic (or Bayesian) graphs, statistical analysis, like correlation coefficients, principal component analysis, factor analysis, and sensitivity analysis using neural networks.
Dependency modeling makes for great preprocessing. However, few methods are available for inferring the structure from data, and they are limited to small databases. Therefore, in general, domain experts have to provide insight in data structures.

Correlation coefficients

The correlation is a measurement of the strength of the linear relationship between two variables. De-correlated data results in a correlation coefficient of 0; equivalent data sets have a correlation coefficient of 1. When using neural networks, a significant correlation between the variables of the input data should be avoided in order to obtain better performance. An often-used technique to de-correlate data is the Principal Component Analysis.

Principal Component Analysis

The idea behind Principal Component Analysis (PCA) is to change N correlated variables into N other, de-correlated variables. The newly de-correlated variables are measuring different dimensions in the data; they do not overlap anymore. The variables are ordered so that the first displays the largest amount of variation, the second variable the second largest amount of variation, and so on. These variables are called the principal components. When the variances of most of the variables are so low as to be negligible, then the variation in the data set can be adequately described by the few variables that are not negligible. A description of the calculations involved in this analysis can be found in [15].

However, PCA does not always improve performance. If the data contains outliers or meaningless variables with a high noise level, as is usually the case with real-life data, then one should be careful before using the principal component analysis.

3.5 Analysis methods applied to power output prediction

For the prediction of the power output of non demand-driven power systems we will use the analysis methods mentioned in the previous sections. This section contains the theoretical application of the different analysis methods to the problem of power prediction: we explain how each method can be used in the process of creating a prediction model from measured data.

First of all, summarization only describes data, so this method can be very useful for data analysis in terms of characterizing the data. Knowledge about the statistics of the data might help in choosing modeling parameters.

Furthermore, the prediction has to be accurate for the next 24 to 48 hours. This time interval is too long for time-series to be able to predict accurately, since they mostly depend on the direct past and then extrapolate, where the error increases pretty fast in time. But we can use the preprocessing techniques to capture the time components in features. By capturing the time components we create the bank of linear filters, the first functional block described in the universal myopic mapping theorem (section 2.4). We will use Fourier transforms and autocorrelation functions for this purpose.

Seen from a mathematical perspective, the second functional block of the theorem has to be a static, functional mapping from multiple inputs to one output, where all data types are continuous. This is a typical regression problem. The relationship between the weather components and the power production of the whole park is nonlinear, so linear regression methods will probably not be able to accurately describe this input-output relation. Because of the lack of understanding of the underlying physical system, it is not possible to specify equations for statistical nonlinear regression models. MLPs are especially useful for function approximation/mapping problems, and we will use them to create the prediction model.

The different weather conditions, for example temperature and sun-radiation, are not independent. Temperature also depends on the time of the day. This makes the data upon which the prediction and thus the model have to be based multivariate data. Multivariate data can be modeled with dependency-modeling techniques. When used as a preprocessing technique, the dependency model provides independent variables as input for the MLP. We will apply the Principal Component Analysis to de-correlate the input data.

Summary

Figure 3-1 depicts the modeling steps involving the data analysis methods of this chapter. We begin the preprocessing phase using summarization techniques to characterize the raw data. After that the features will be extracted. In the last step dependency-modeling techniques are employed to de-correlate the features. For the actual prediction we will use MLPs as a regression model.

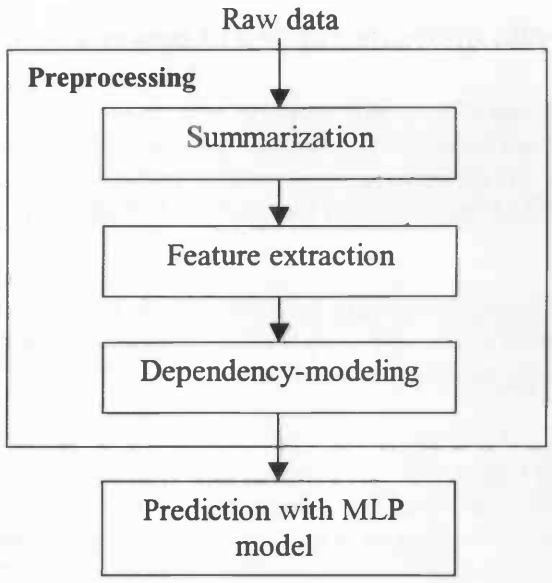

Figure 3-1 Modeling steps involved in preprocessing and prediction

The steps of Figure 3-1 are an important part of the process of creating a prediction model. The theoretical background of the entire modeling process can be found in Appendix A.

3.6 Results

This section contains the results of the application of the preprocessing steps of Figure 3-1 to real-life data. We will start with a description of the data. The actual prediction modeling of the data will be discussed in Chapter 4.

3.6.1 Data

For the modeling of a real-life problem we will use historical data from a park of 116 installations, located in Gelderland, the Netherlands (region 1). A model will be build based upon the hourly weather measurements for humidity, sun radiation, temperature, wind direction and wind speed, and on the measured power production in kW of every 15 minutes of each branch of the whole park, for the year 2001. The weather data was adjusted to the 15-minute time scale required by TenneT, by giving the quarters the same value as the corresponding whole hour. Since the year 2001 has 365 days, a day 24 hours, and an hour 4 quarters, the data consists of $365*24*4 = 35040$ patterns.
The weather data we will use consists of weather measurements instead of weather forecasts, because forecast data is not available. Test results will be based upon the assumption that weather measurements are a suitable representation of weather forecasts. A second data set of another region (region 2) in the Netherlands contains hourly weather and production measurements of the first three months of 2001.

When the model is in use, there will be weather forecasts from the weather company Meteo Consult, with hourly data for humidity, sun radiation, air temperature, wind speed, and wind direction.

The model is to be based on the behavior of the whole park, and the model output will be the predicted power output as fraction of the total nominal power of the park. This way the model does not have to be altered whenever individual systems are added or removed.

Table 3-1 and Table 3-2 give a summarization of the data. The installations are divided into branch A and the remaining branches, B. This is because the A branch accounts for 53.7 % of the data, while the number of installations from other branches is much smaller. The production is given as a fraction of the total nominal power per branch.

	Min	Max	Mean	Std	Range	Kurtosis[1]	Skewness[2]
Humidity (%)	23.00	100.00	83.47	15.11	77.00	3.8156	-1.1845
Sun Radiation ($J/cm^2/h$)	0.00	345.00	40.60	67.56	345.00	6.0717	1.9237
Temperature (deg C)	-10.10	31.70	10.10	7.17	41.80	2.5368	0.0965
Wind Direction (deg)	0.00	360.00	187.50	94.48	360.00	2.1812	-0.2696
Wind Speed (m/s)	0.00	13.00	3.79	2.12	13.00	3.0639	0.5258
Production A (P/Pn)	0.00	0.79	0.32	0.20	0.79	3.0639	0.2221
Production B (P/Pn)	0.01	0.78	0.32	0.22	0.77	1.5986	0.2238

Table 3-1 Region 1 2001

	Min	Max	Mean	Std	Range	Kurtosis	Skewness
Humidity (%)	57.00	99.00	88.47	8.02	42.00	4.2306	-1.2030
Sun Radiation ($J/cm^2/h$)	0.00	195.0	18.30	32.59	195.00	7.5535	2.1562
Temperature (deg C)	-7.3	12.60	3.88	3.73	19.90	2.6929	-0.0053
Wind Direction (deg)	2.00	360.00	175.07	91.64	258.00	2.0975	0.0688
Wind Speed (m/s)	1.00	11.00	4.56	2.01	10.00	2.3633	0.2397
Production A (P/Pn)	0.30	0.87	0.68	0.10	0.57	3.2090	-0.8357
Production B (P/Pn)	0.25	0.75	0.57	0.09	0.50	2.6871	-0.5902

Table 3-2 Region 2 January-March 2001

[1] *Kurtosis* is a measure of how outlier-prone a distribution is. The kurtosis of a distribution is defined as

$$k = \frac{E(x-\mu)^4}{\sigma^4}$$

where μ is the mean of x, σ is the standard deviation of x, and E(t) represents the expected value of the quantity t.

[2] *Skewness* is a measure of the asymmetry of the data around the sample mean. The skewness of a distribution is defined as

$$y = \frac{E(x-\mu)^3}{\sigma^3}$$

where μ is the mean of x, σ is the standard deviation of x, and E(t) represents the expected value of the quantity t.

3.6.2 Summarization

Summarization describes data in terms of statistic features of the data.
Part of the summarization can be found in tables Table 3-1 and Table 3-2. The ranges of the various variables differ in size, which is something neural networks have difficulties with. Scaling the data between –1 and 1, or normalizing with zero mean and standard deviation of 1 are standard solutions for this.

A common assumption in many time series techniques is that the data is stationary. A stationary process has the property that its statistical characteristics do not change over time, and that there are no periodic fluctuations.
Due to its nature our data is highly periodical. Yearly and daily periodicities exist in the weather conditions, like the average temperature throughout the year or the incoming sun radiation during the day. One would expect these periodicities in the power output data as well, since the power output is highly influenced by the weather conditions.

Figure 3-2 shows the mean power fraction of each day of data set A. The mean power fraction lies between 0 and 0.2 in the summer, while it mostly lies between 0.5 and 0.7 in the winter months; the mean of the power output obviously changes over the year. The mean power fraction of data set B follows a similar pattern. So the power output is clearly non-stationary, and its trend is more or less opposite to the mean temperature during the year. When we would have had data from multiple years, we would probably see this pattern re-occurring every year. This kind of periodicity is called seasonality.

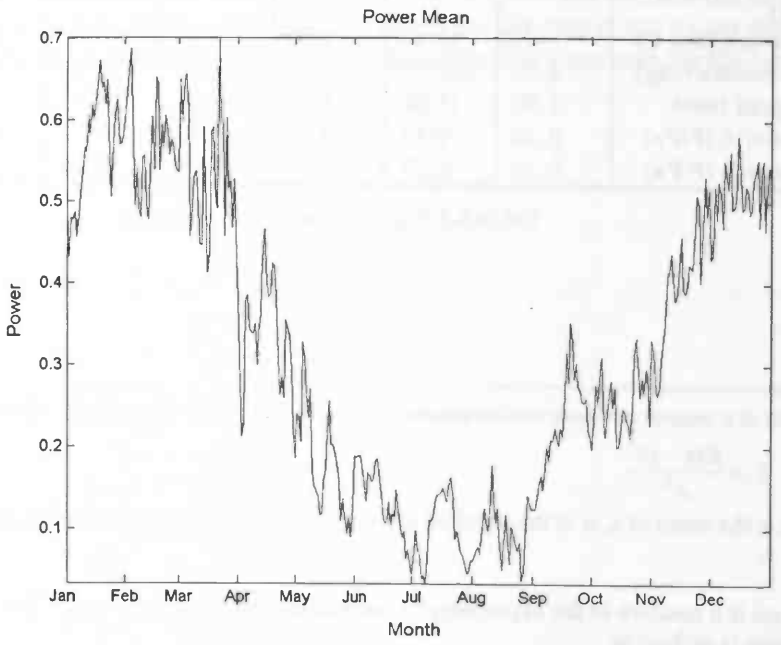

Figure 3-2 Mean power output over the year of data set A. The mean power output is high in the winter and low in the summer months.

The data show daily periodicity as well. In the mean weekly pattern in Figure 3-3 the 24-hour period is unmistakably evident. This non-stationarity is time-dependent and deterministic.

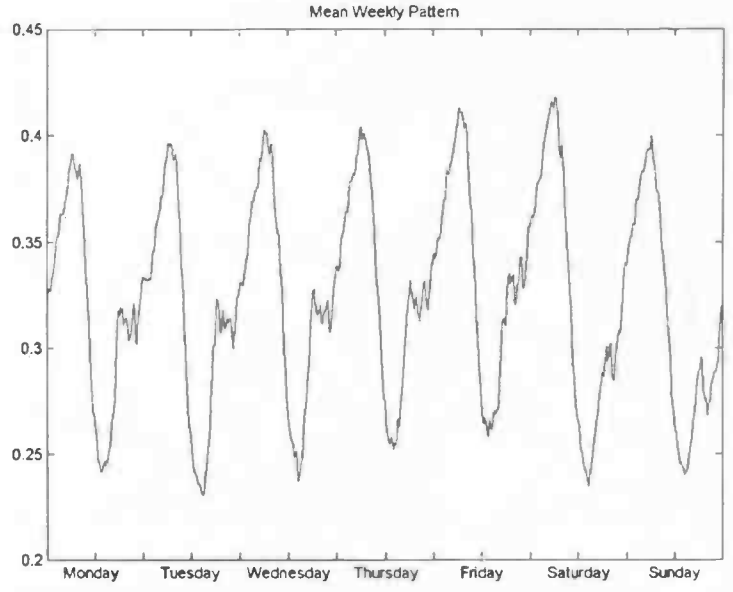

Figure 3-3 Mean weekly pattern of data set A. The different days of the week are separable and have a similar pattern.

So not only the weather conditions in the data show non-stationarity. From both Figure 3-2 and Figure 3-3 we can conclude that the power output is non-stationary as well. Since the time and the trends in the weather conditions determine the trends in the power output, the non-stationarity can be modeled well.

Depicted in Figure 3-4 and Figure 3-5 are the mean daily patterns per three months of the power output for the two data sets. The difference in pattern between the two sets is clearly visible. The pattern of branch A shows an early morning peak, afternoon valley and then rises again for the evening, while the other data set has a low production at night and a high production during the day, without those strong peaks. Because the production patterns are so different, separate neural networks will be trained for the two subsets.

In Figure 3-4 not only the amount of power, but also the shape of the pattern changes per three-month period. The winter months have much stronger peaks and valleys than the smooth summer months. This is a good indication that we need a different neural network for the different periods as well. Other tests have shown that no two months have the exact same pattern. But the changes in patterns grow smaller when the time periods are smaller, so at some point they will not be significant enough for it to be useful to train different neural networks on. It might also be useful to train different networks on the peak and valley hours, since they have such different characteristics.

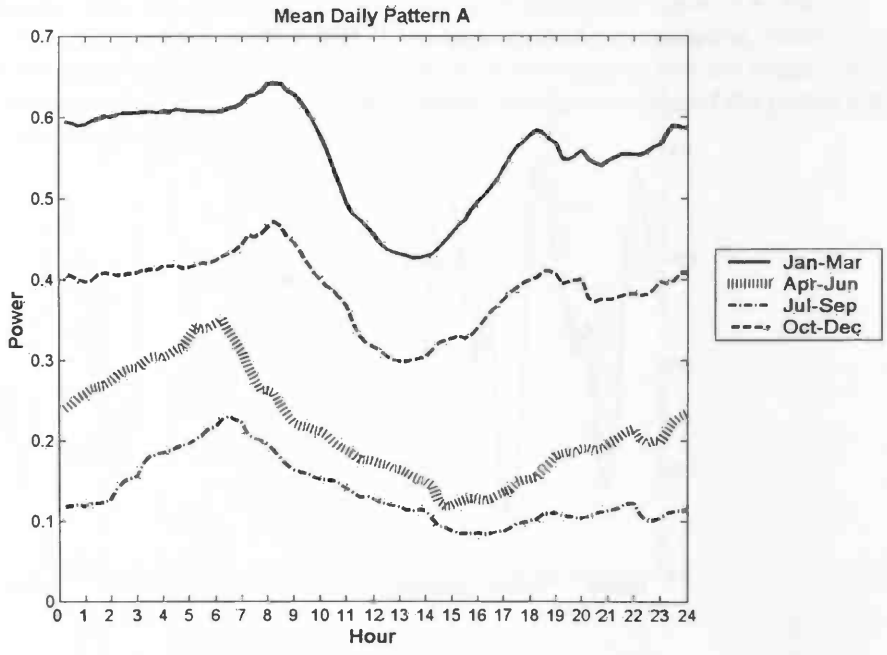

Figure 3-4 Mean daily power per season of data set A. Per season the mean amount and pattern of the power changes. The lowest mean power output is from July to September, and the highest from January to March.

The shift in the amount of power and pattern shape of data set B can be seen in Figure 3-5. These pattern changes are not as big as can be seen in the A data set. But the afternoon output in the summer months shows a valley, while this is not the case in the winter months. Similar to data set A, these differences in patterns decrease when making the time periods smaller.

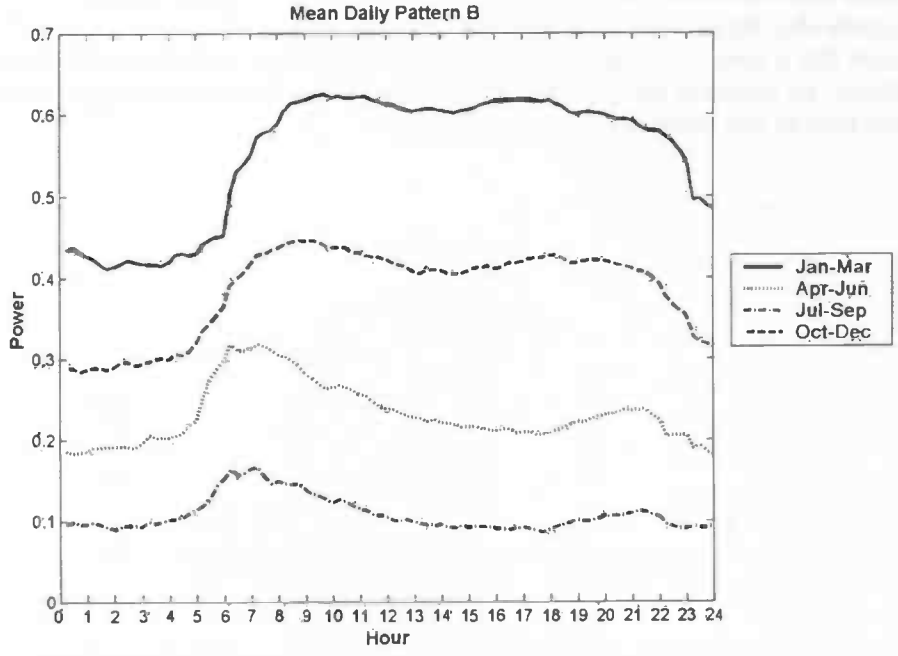

Figure 3-5 Mean daily power per season of data set B. Per season the mean amount and pattern of the power changes. The lowest mean power output is from July to September, and the highest from January to March. The patterns of this data set are clearly different from the patterns of data set A.

Other tests showed that data set B displays a large difference between weekday patterns and weekend day patterns. It might be useful to separate these days and build different models for them, or include the mean patterns as an extra input variable for the neural networks. We will return to these issues in section 4.2.2.

The correlation coefficients of the correlation between four of the available input variables and the power output can be found in Table 3-3. The sign signifies the direction of the correlation and the absolute value signifies the strength. The strongest correlation exists between the temperature and power output, while wind speed has the weakest correlation with the power output. The strong linear correlation between temperature and the power output demonstrates an important linear relation between the two variables. Note that the correlation between the wind direction and the power output is not included. The wind direction is measured in degrees and not linear itself, so its correlation coefficient with the power output would be meaningless.

Humidity	Sun Radiation	Temperature	Wind Speed
0.4800	-0.3900	-0.8900	-0.0100

Table 3-3 Correlation coefficients of input variables and the power output

When we plot the hourly temperature values against the fractional power output, we expect to see this linear relationship.

Each of the 24 dots in Figure 3-6 and Figure 3-7 represent a mean hourly value of the temperature and the fractional power output over 2001 of data sets A and B respectively. The curved arrow signifies the direction over the 24 hours of a day. Both shapes indeed show the

linear relation between the two variables, which is most evident in Figure 3-6. The two figures also depict a nonlinear relationship in the form of the loop and curve. Further studies show that a nonlinear relationship exists between all input variables and the output variable. Hence, we conclude that nonlinearity is characteristic for the modeling of the power output of this kind of non demand-driven power systems.

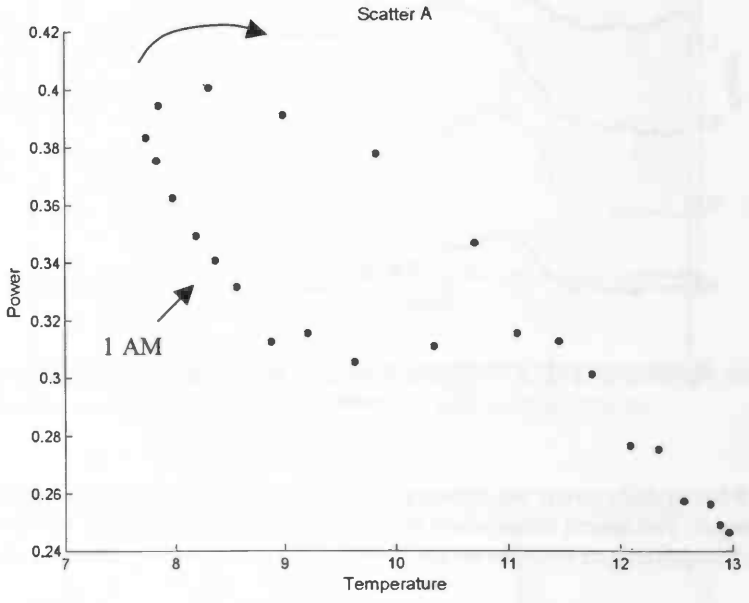

Figure 3-6 Mean hourly temperature and fractional power output over 2001 of data set A. There is a clearly visible nonlinear relationship between the two variables in the shape of the loop at the upper-left part of the graph.

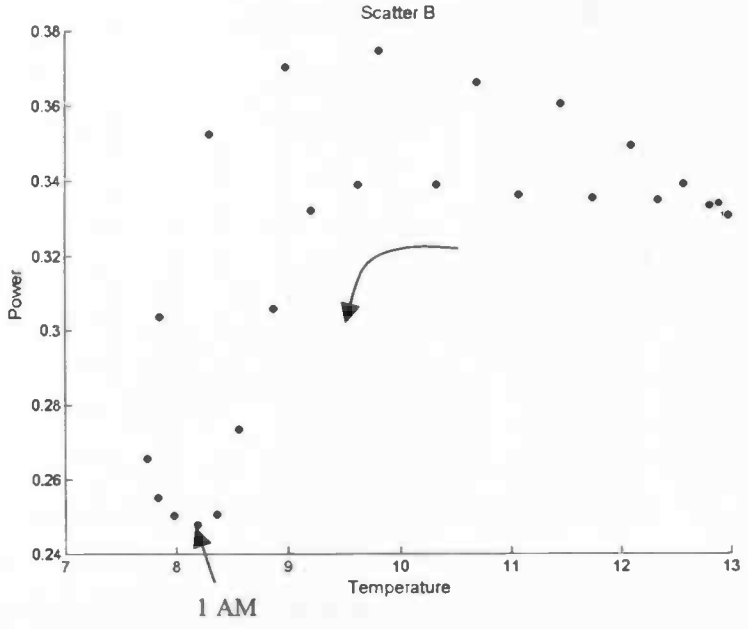

Figure 3-7 Mean hourly temperature and fractional power output over 2001 of data set B. The nonlinear relationship is evident by the loop in the structure of the dots.

3.6.3 Feature extraction

The Fourier transform of the power output of data set A is depicted in Figure 3-8, and on a larger scale in Figure 3-9. Figure 3-8 shows a very strong peak at 24 hours. So this is an important period in the power output. This was to be expected; the mean day patterns also showed a daily period. Unexpected are the smaller peaks at 12, 8 and even 6 hours in Figure 3-9. These periods were also found in the B data. They originate from the way the users employ their power installations and from the periodicity in weather parameters.

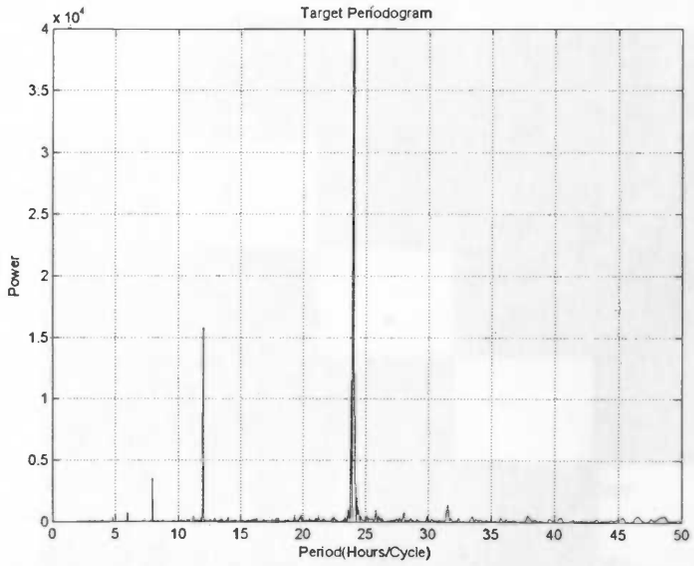

Figure 3-8 Fourier transform of the power production of data set A. The largest peak corresponds to a 24-hour period in the data. Smaller peaks can be seen at 8 and 12-hour periods.

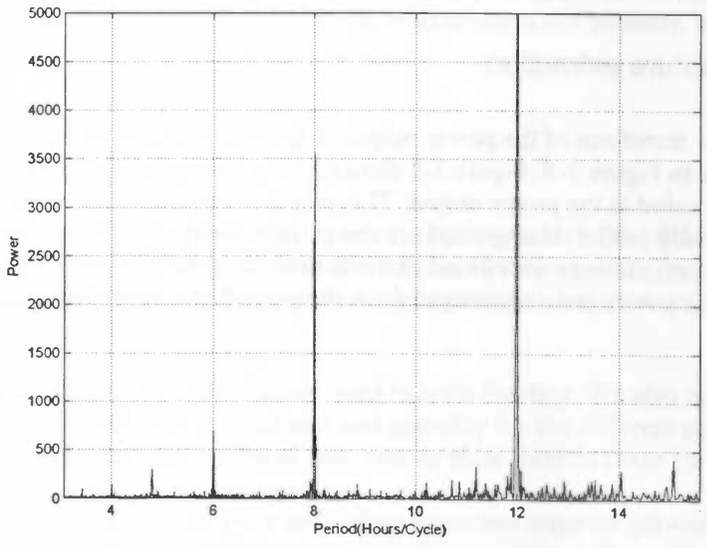

Figure 3-9 Zoom of the Fourier transform of the power production of data set A. The 6, 8, and 12-hour periods are depicted by the three largest peaks.

Figure 3-10 shows the autocorrelation of data set A of region 2. The x-axis scale is a quarter of an hour. The small peaks represent the 12-hour periods and the large peaks the 24-hour periods that are also revealed by the Fourier transforms. The stronger peaks at 4 and 8 days demonstrate the existence of a 4-day period. The autocorrelation function was also applied to the other data sets, where periods of 2, 3, 4, and 7 days were visible.

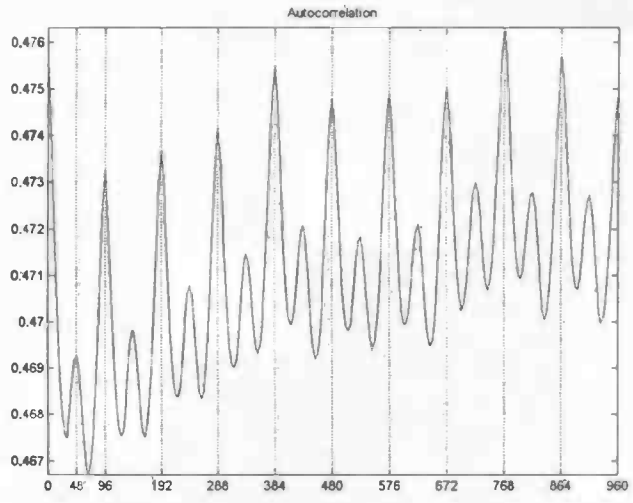

Figure 3-10 Autocorrelation of data set A of region 2. The x-axis scale is a quarter of an hour. A small peak arises at every 48 quarters, and large peaks at 96 quarters, the 12-hour and 24-hour periods respectively. Stronger peaks at 384 (4 days) and 864 (8 days) demonstrate the existence of a larger period of 4 days.

By applying the Fourier transform and autocorrelation function several features have been discovered. The most important periods in the data are:

- Hourly: 6, 8, 12, and 24 hours
- Daily: 2, 3, 4, and 7 days

But at the time for which the power output is predicted, no measurements of the actual power output from at maximum 48 hours ago are available to base the prediction on. Therefore, for this period we will extract the weather forecasts as extra features.

All these important features will be extracted when preprocessing the data.

3.6.4 Dependency Modeling

Figure 3-11 shows the correlation between the different weather variables. The white areas represent a 100% correlation, and the black 0%. The gray values represent values in between. The significant correlation between humidity and sun radiation is clearly visible, while there is (almost) no correlation between humidity and the wind direction.
Correlations between the variables of the input data should be avoided in order for neural networks to get a better performance. Tests showed that the application of the Principal Component Analysis to the input data yields an overall improvement in error performance of 7 %. Therefore we will use the Principal Component Analysis to de-correlate the data.

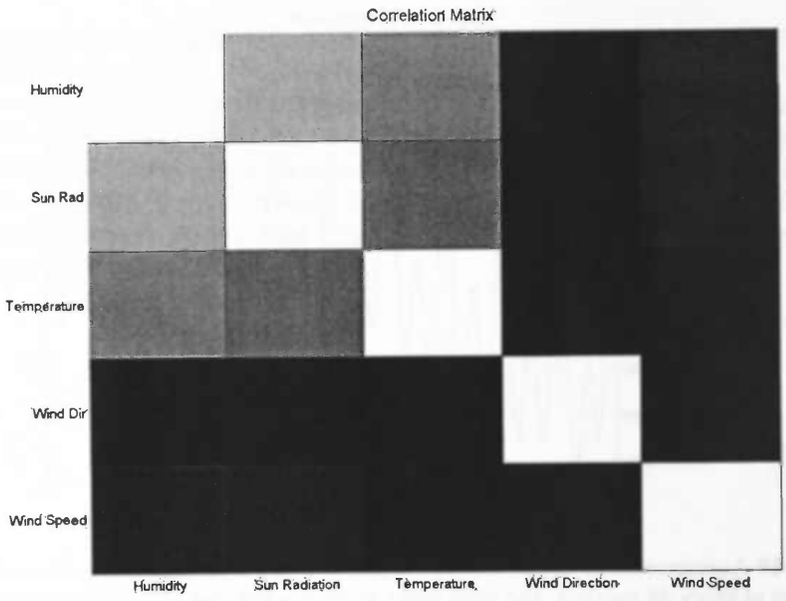

Figure 3-11 Correlation plot of the weather components humidity, sun radiation, temperature, wind direction, and wind speed. Light areas signify a strong correlation between two variables, and dark areas a weak correlation. These correlation strengths, like the strong correlation between temperature and sun radiation and the weak correlation between wind direction and humidity, are normal weather correlations.

3.6.5 Conclusion

The data analysis has discovered some important characteristics of the data of the real-life problem. Note however that all these characteristics are problem dependent. When building a model for a different problem, these tests have to be performed again to extract that data's characteristics.

From the summarization we learn that we need to scale the data. We also need two different networks for the two different data subsets and probably for the different periods of both the subsets as well. These periods can be of one, two, or three months (case 1).

Separate neural networks for the peak and valley hours and separate networks for the week and weekend days might also result in a better performance (case 2).

The data has to be preprocessed to de-correlate the different variables by using principal component analysis. More preprocessing is needed to extract the important features that the sequence analysis revealed.

Description of the prediction system
The complete prediction system for each data subset will be based on a selection of the most relevant input variables for the specific subset. The total amount of available input variables is now 52 and consists of the following categories:

	Category	Components
1	**Time Factors**	Time of the day Day of the week Summer/wintertime bit
2	**Measured produced power**	History of the past 2, 3, 4, and 7 days
		Average value over several periods in the past, of which the owner of the installations indicated them as being important
		Average day pattern and average week pattern for the moment of the prediction
3	**Weather factors** - Humidity - Sun radiation - Temperature - Wind speed	Forecast for the moment of prediction
		History of the past 6, 8, 12, and 24 hours
		Average value over several periods in the past, of which the owner of the installations indicated them as being important

Table 3-4 Available input variables for the prediction system in use

The prediction system itself consists of a PCA conversion and a neural network model. After the PCA conversion the selection of relevant input variables is directly fed into the network. The single output of the network is the prediction of the fraction of the power output in kWh for one quarter of an hour.
Let *h* denote one hour and *d* denote one day. Then the prediction system for the prediction at time *k* is as depicted in Figure 3-12.

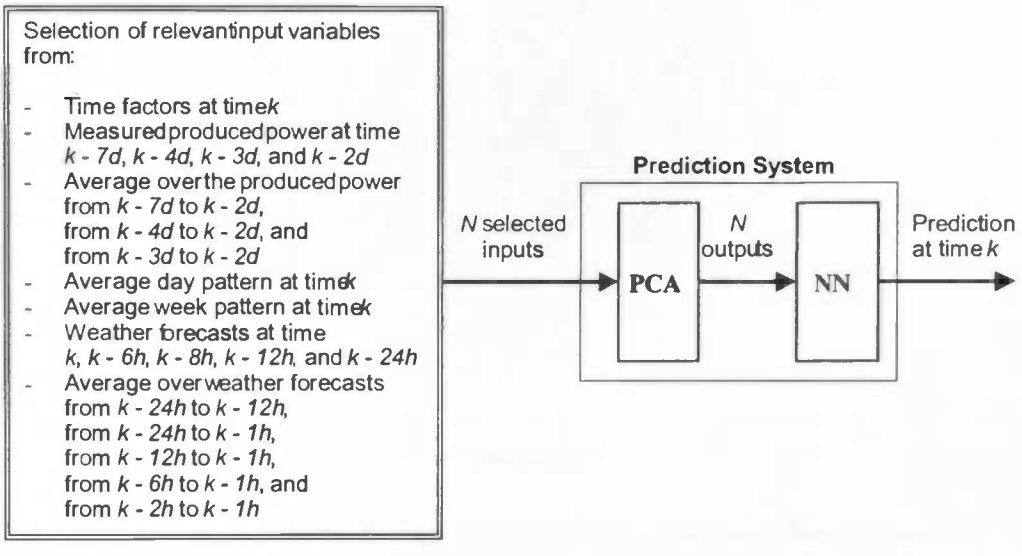

Figure 3-12 The prediction system. The prediction at time *k* is based on measurements from one week before *k* to forecasts at time *k*. The selected inputs are converted by the principal component analysis before they are fed into the neural network for the actual prediction.

The difference between Figure 3-13 and Figure 3-14 show the improvement of the predictability resulting from extracting extra features as input variables. Figure 3-13 depicts the first two principal components of the input variables before feature extraction on the two axes, and the power output as the colored z-value. The colors overlap a great deal, so the predictability of the power output based on the first two principal components is bad. Figure 3-14 shows the relation between the first two principal components and the power output after the features are extracted. The power output is now almost linearly separable, which means a better predictability.

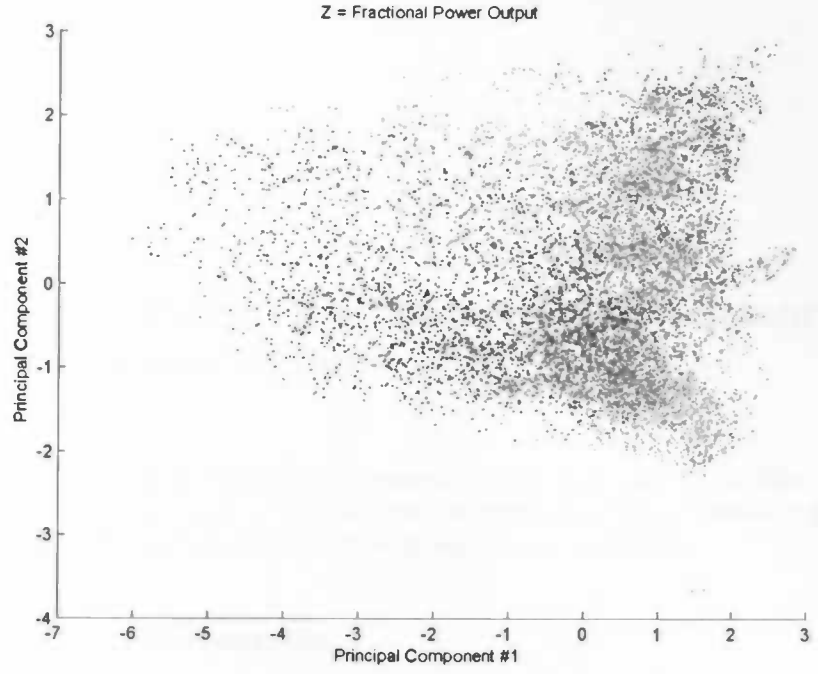

Figure 3-13 Scatter of first two Principal Components and the Fractional Power Output of the combined data sets before feature extraction. The colors overlap a lot, which means that the predictability of the power output is low.

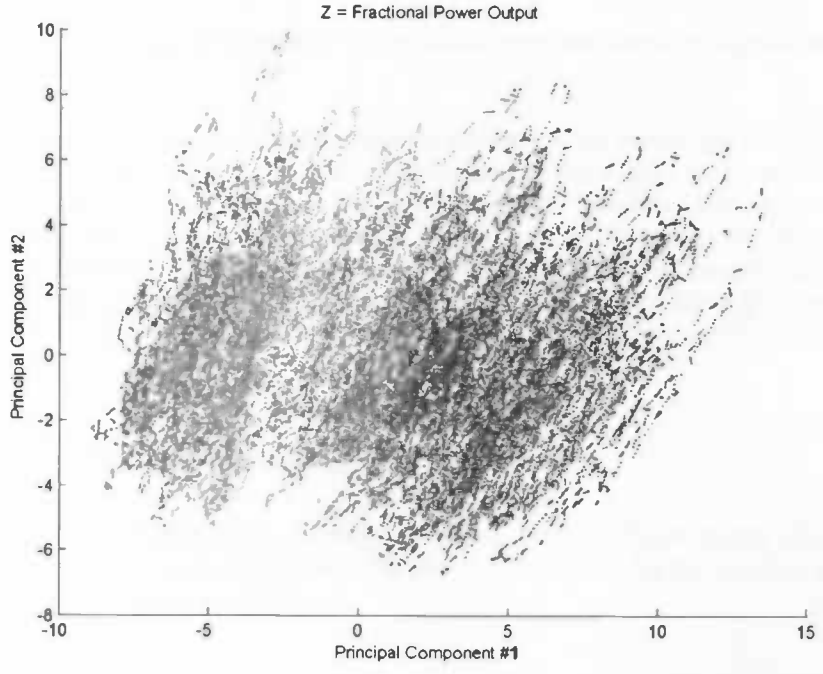

Figure 3-14 Scatter of first two Principal Components and the Fractional Power Output of the combined data sets after features extraction. The colors are almost linearly separable, which means that the predictability of the power output is high.

Chapter 4 Output prediction for a non demand-driven power system

This chapter contains the results of the research on the actual output prediction of non demand-driven power systems. We first give a description of the performance measures we used. After that the performances of different models are compared.

4.1 Performance Measures

The neural networks are compared by their root mean square error (RMSE) of the target t_i and the network output on the test set o_i, where N is the number of patterns:

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(t_i - o_i)^2}$$

The RMSE is the standard deviation of the model error, and therefore a good indication of the spreading of the error.

As a second performance measure the imbalance costs of the prediction error can be taken. If the predicted power output is different from the actual output, then the owner has to pay TenneT for the difference; these are the imbalance costs. Imbalance costs vary along with price fluctuations on the electricity market. The study of M. de Noord showed that an average of 4 Euro per MWh is a fairly good estimate of these costs. But because this is an estimate, and because the real prices change constantly, they do not really offer a good prediction error measure. Therefore, we will just use this measure as a final indication of the imbalance costs of the prediction error.

4.2 Modeling and Predicting Power Output

In this section the results on the training and testing of the MLPs are given. The networks are trained by using the Levenberg-Marquardt training algorithm. For the creation of a train and test set we split the data randomly.

Model selection

The experiments presented in this section are preceded by model selection tests to decide on the optimal number of hidden neurons, given the different data sets and the input variables. Here we define the optimal number of hidden neurons as the number with which the networks have the best performance on the test set, and do not start overfitting the training data after some number of epochs.

This technique to obtain a good generalization performance proves to be better on our data than the early stopping method, where the number of hidden neurons is large and training is stopped as soon as the error on a validation set start increasing. Using the early stopping method on data that was split periodically in 24-hour periods, instead of using the method on data that was split randomly, results in a worse error performance as well, see also Figure 4-1. Unfortunately we were not able to investigate the reason for this difference in performance.

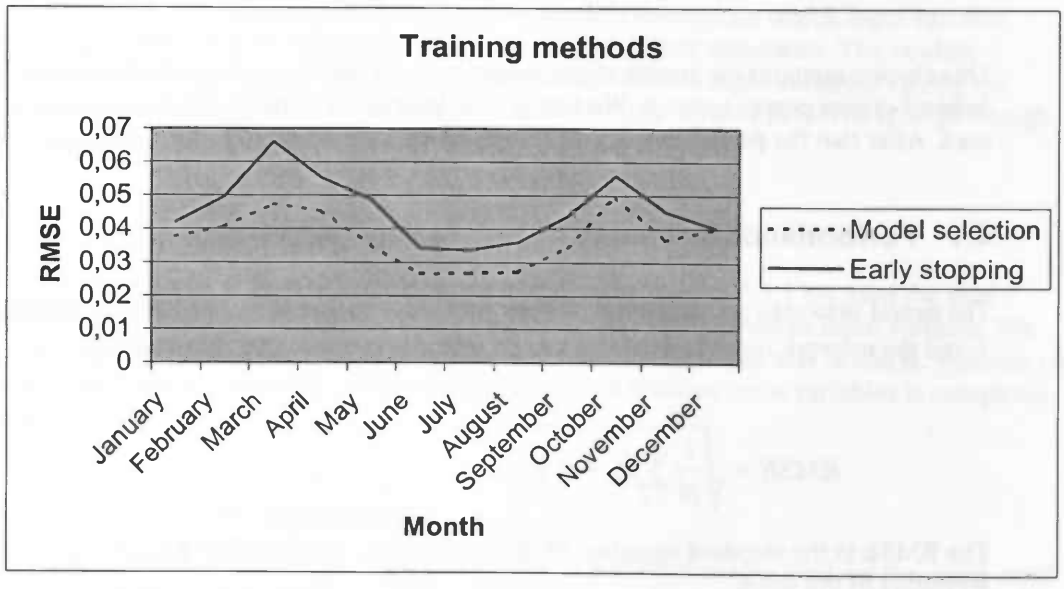

Figure 4-1 Average performance of 100 neural networks trained on monthly periods of data set A. The RMSE values of networks trained with the model selection method is depicted as a dotted line, networks trained with the early stopping method is depicted as a solid line.

Next we will discuss the results from the different relevant input determination methods. The section after that contains the test results constituting the two cases from the conclusions of section 3.6.5. Finally the best type of networks is used to create ensembles, of which the RMSE results are shown.

4.2.1 Relevant Input Variables

The 52 input variables available for training do not all have to be important. To reduce the input dimension and so obtain better generalization results, we only want to use the most relevant input variables for each period in which the data is split up. This means that neural networks trained on different data sets can have different input variables.

To determine the relevant input variables, we used the complete set of available input variables. The neural networks we trained and tested all have one hidden layer. A total of 24 data sets were used for training, 12 periods of one month for both the A and B data sets. Each data set was randomly split into a training and test set.

Input selection with partial retraining

The most promising input relevance determination algorithm discussed in Chapter 2, input selection with partial retraining, turned out to be highly unstable on our data set. The order of relevance of the input variables for each of the 24 data sets was not reproducible, even when we trained and tested neural networks on the same split of training and test set.

But we are interested in the subset of most relevant inputs, and not necessarily the order of relevance. We tried to be able to retrieve all the inputs that have a high relevance for multiple neural networks trained on the same data set, by extending the input selection algorithm with a simple weighting scheme. After every run of the algorithm, the resulting input relevance list is weighted according to the relative position p of each of the N inputs in the list. Weight α_x of input x is:

$$\alpha_x = \frac{p_x}{\sum_{i=1}^{N} p_i} \quad \text{with} \quad \sum_{i=1}^{N} \alpha_i = 1 .$$

The different weights of each input variable x, produced by M runs of the algorithm on M neural networks trained on the same data set, are added together for the new position p of x

$$p_x = \sum_{m=1}^{M} \alpha_x^m .$$

Finally, sorting the positions of all the input variables forms the new input relevance list L_{new}.

Table 4-1 contains as an example of the performance of this input relevance algorithm the results of ten neural networks. Each network was trained and tested on the month June of data set A. The second column of the table holds the most relevant input of each of the ten networks, according to the input relevance with partial retraining algorithm. We included only the most relevant variable for each network to keep the example simple. The position in the new input relevance list L_{new} of each of these variables is given in the third column.

Network	Most relevant input	New position in L_{new}
1	Mean temperature over the past 1-24 hours	20
2	Wind speed (current time)	22
3	Wind speed 6 hours ago	27
4	Temperature 6 hours ago	38
5	Humidity 24 hours ago	12
6	Humidity 24 hours ago	12
7	Temperature 24 hours ago	7
8	Mean humidity of the past 1-24 hours	11
9	Mean wind speed over the past 12-24 hours	19
10	Mean temperature over the past 24-12 hours	44

Table 4-1 L_{new} June data set A

Unfortunately, none of the most relevant inputs of the ten neural networks in Table 4-1 are among the most relevant 6 inputs of L_{new}; each of the variables, denoted as the most relevant by one or more of the 10 networks, has position 7 or lower in L_{new}. We encountered similar results when using the algorithm for the other data sets. Every time the relevant input variables of the single networks became irrelevant when they were combined according to

their weights. Neural networks trained on the most relevant inputs according to L_{new} performed worse than the single networks trained on all the input variables.
The explanation for this phenomenon is as follows. The most relevant input variables of the single networks come from several categories. In our example these categories are temperature, wind speed, and humidity. For this example each component of these categories essentially holds equally important information about the power output. So during the training process the neural networks arrange their synaptic weights in such a way that the signal from a component of one of these categories is more amplified than any other signal. This input variable is the most relevant input. The other input variables from these categories are now considered irrelevant since their removal from the neural network does not degrade the error performance; the same information is already extracted from the signal of the most relevant input variable. The random initialization of the weights determines which input variable comes up as the most relevant, since that is the only changing parameter. The random initialization results in a uniform distribution of most relevant input variables among the components of these categories, as long as the number of networks involved is large enough, and they eventually become irrelevant in this weighting scheme.

Linear input relevance determination

For the linear input relevance determination described in section 2.5.1 we used the stop criterion of a minimum RMSE decrease of 0.0005 when adding another input variable. We determined the relevant inputs this way for each month of both data sets A and B. Because no neural networks are involved, the resulting selection of relevant input variables is completely reproducible.

Naïve input relevance determination

The stop criterion of a minimum RMSE decrease of 0.0005 was also used for the naïve input relevance determination (section 2.5.2). For each subset of an equal number of input variables we trained five neural networks, each on a different bootstrap of the same monthly data set. Subsequently, we compared the performances on the subsets using the average RMSE of the five networks. The most relevant input variable subset is the one resulting in neural networks with the lowest average RMSE on the test set. The adding of input variables to the previous most relevant subset was stopped when the stop criterion was reached and the subset contained three or more input variables. A minimum of three input variables was necessary because the RMSE did not start decreasing until the networks were trained on a minimum of three input variables.
The resulting selection of most relevant inputs is almost completely reproducible, in contrast to input selection with partial retraining. Of a total input variable selection of, on average, nine input variables for all data sets, only the last one or two input variables added changed for some runs.

To compare the error performance of networks trained on the inputs selected by the linear and naïve input relevance determination algorithms, we trained 100 neural networks for each input variable selection for every month of both data sets. Each of these 100 networks were divided in four times 25 networks with each 2, 4, 6, or 8 hidden neurons. The RMSE values were then averaged over all 100 networks over data set A and B together. This resulted in 12 average RMSE values for both input relevance determination algorithms.

The RMSE values for each period are depicted in Figure 4-2. The error performance of neural networks trained on input variables selected by the linear input relevance determination is slightly better for most of the periods. But the differences in RMSE between the two methods

are so small that they can be caused by the random split in training and test set of the data, and the random initialization of the synaptic weights of the neural networks. When the most relevant variables to determine the underlying regression function coincide with input variables that have the strongest linear relation with the output variable, then it would not matter which input relevance determination algorithm is used. The instability of neural networks might then cause the naïve method to select a subset of most relevant input variables a little less true to the underlying problem than the linear method.

Because the linear input relevance determination algorithm is faster than the naïve algorithm and the error performance is at least similar, we used this algorithm to determine the most relevant inputs when training neural networks for other tests in this research. We will return to this issue when we consider the imbalance costs of prediction errors as a measure for performance.

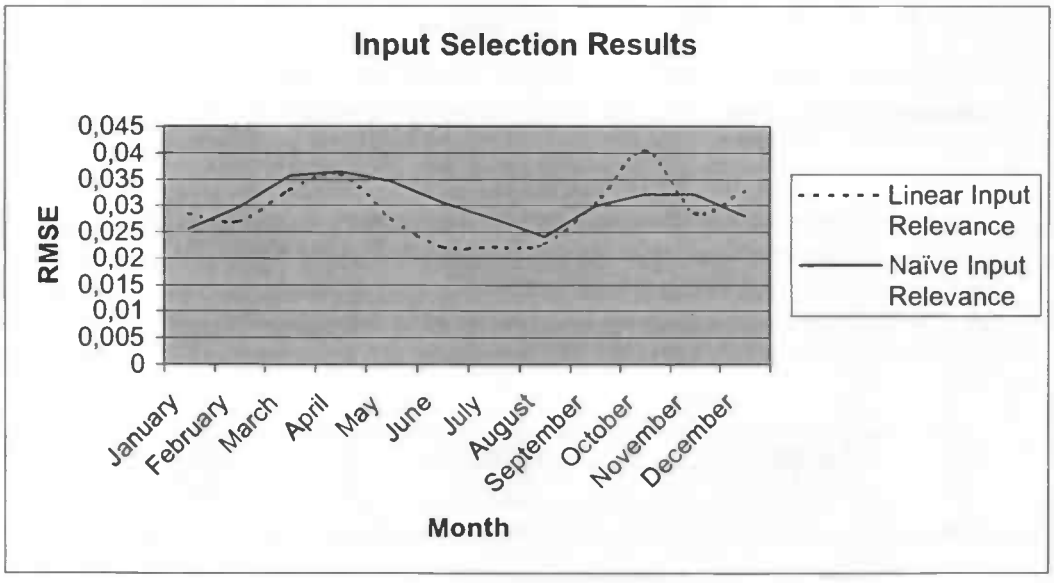

Figure 4-2 RMSE values of networks trained with inputs selected from the linear input relevance determination (dotted line) and the naïve input relevance determination (solid line). The RMSE values are averaged over results from data set A and B together. The RMSE resulting from the linear input relevance is on average practically equal to the RMSE resulting from the naïve input relevance.

4.2.2 Varying the data set size

Case 1

To decide on the size of the periods on which the neural networks should be trained, we compared neural networks trained on data from 1 to 3 months. This is because test results showed that networks trained on four or more months performed significantly worse than smaller periods. We created four periods of three months: January-March, April-June, July-September, and October-December, and six periods of 2 months: January-February, March-April, etc.
For every data period we trained 100 MLP neural networks. All the networks, including the ones trained on multiple months, were evaluated on the same test data set of one month, which was not used in training. This way all the periods could be compared to each other.

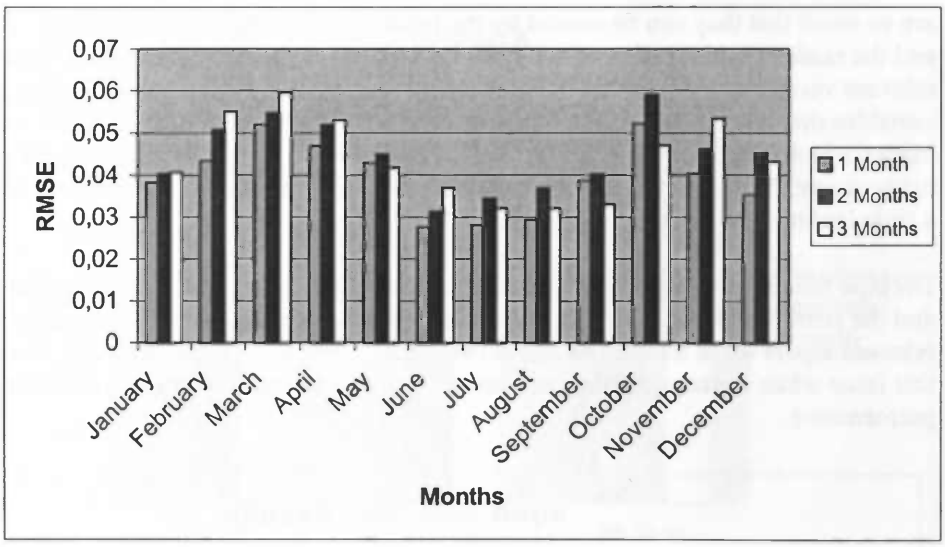

Figure 4-3 RMSE values of branch A. Except in September and October all the 1-month networks have the best performance.

Figure 4-3 shows the RMSE of the best neural networks for data set A. Besides September and October the neural networks trained on monthly data show the best performances. Apart from these exceptions, this is what we expected, since the mean daily patterns even vary for periods as small as one month, as discussed in 3.6.2. Table 4-2 shows that on average indeed monthly data sets result in the best RMSE values.

1 Month	2 Months	3 Months
0.0457	0.0485	0.0483

Table 4-2 Best RMSE values of branch A averaged over the year. The 1-month periods have the lowest average RMSE.

For the two exceptions, the networks have better performances when trained on three months of data. This is probably because September has a pattern equal to the other two summer months of the training data: July and August, and October equal to the autumn months November and December. Because multiple months sometimes give better results, we expect that ensembles of networks trained on different periods have a better performance than the ones trained on only one month.

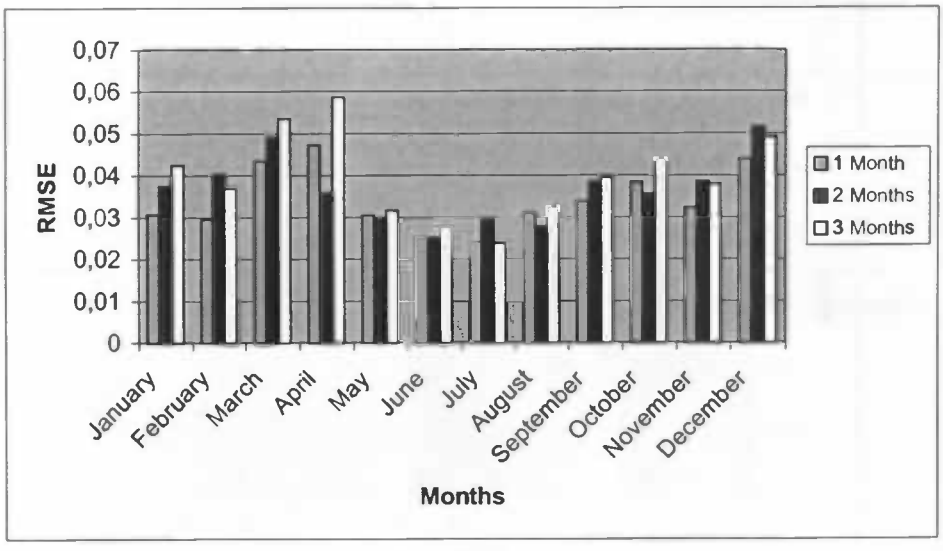

Figure 4-4 RMSE values of branch B. For the majority of the months the 1-monthly networks have the best error performance.

In Figure 4-4 the performances of the networks trained on the B data set are on average only slightly better when monthly periods are used, see also Table 4-3. This can be explained by the dissimilarity of utilization of the power installations in the different branches: the A branch has a stronger seasonal dependency. So the expectation about ensembles becomes even stronger. We will return to this issue in section 4.2.3.

1 Month	2 Months	3 Months
0.0400	0.0404	0.0443

Table 4-3 Best RMSE values of branch B averaged over the year. The 1-month periods have a slightly lower average RMSE than the 2-month periods.

Case 2

In order to see whether or not the separation of week and weekend days and peak and valley hours would result in better performances, we trained and tested neural networks on three different periods: weeks 4-13, 22-31, and 40-49. These weeks are spread throughout the year and therefore considered as a representative sample. For each data set again 100 networks were trained.
As can be seen in Figure 4-5, these separations do not give a better performance for the A data set. The average week pattern is already included in the inputs of the network, so this can explain why the separation in week and weekend days does not give better results. Another reason, also valid for the peak/valley separation, is that the separation of the data creates too small training and test sets for the neural networks to be able to generalize well; the network is overfitting. But when the data set is made larger by adding more weeks the difference in patterns causes worse results.

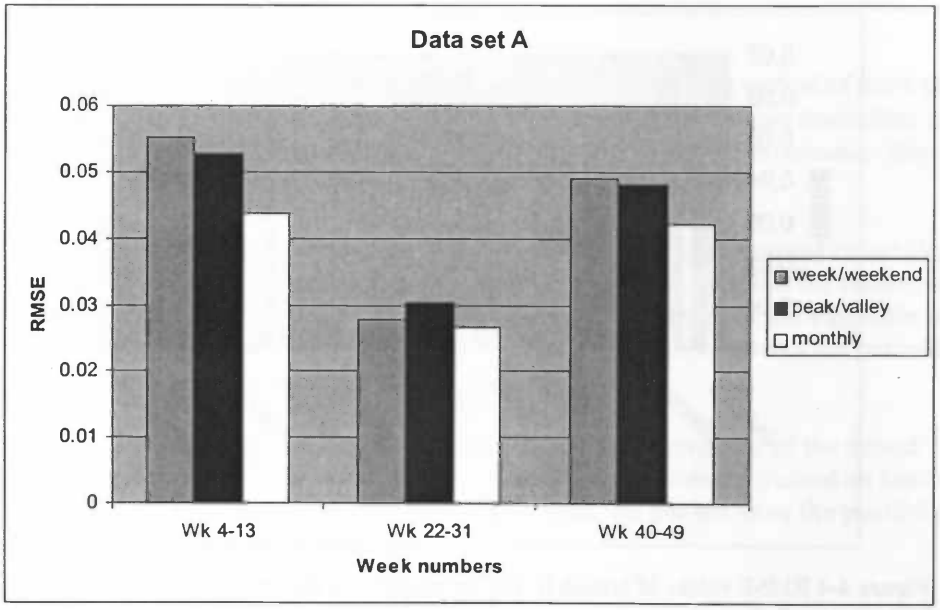

Figure 4-5 Error performances of neural networks for three periods of data set A: week 4-13, week 22-31, and week 40-49. Each column represents the RMSE value of the best performing networks for each period. Separation of the data into week and weekend days or peak and valley hours does not increase performance.

Test results showed that there was a similar increase in error for the peak/valley separation of the B data set. The differences between the week/weekend separation and monthly periods are not as big as in Figure 4-5. Earlier, in section 3.6.2, we saw that the pattern of these days varies more in the B data set, so this is no surprise. But the performance of the networks trained on monthly data still is better.

Thus, the best error performances were reached with neural networks trained on data sets of periods of one month, without a peak/valley or week/weekend separation. These optimal 12 data set periods were used for the creation of ensembles of neural networks.

4.2.3 Predicting with ensembles

For the ensembles we linearly combined 50 networks, each trained on bootstraps of the same data set. This way each network in the ensemble is trained and tested on a different train and test set respectively. We used the balancing method described in chapter 2 for the combination of the networks. The error performance of these 12 ensembles can be found in Figure 4-6 for data set A, and Figure 4-7 for data set B. Next to each ensemble entry in the figure is the average performance of the 100 single neural networks, trained and tested on the same data sets. In both figures the ensembles have significantly lower RMSE values for every period, so ensembles indeed produce better results than the average single network.
It should be noted though, that the single neural networks with the lowest RMSE value per period have a lower RMSE than the corresponding ensembles. However, this does not necessarily mean that these single networks have a better generalization performance. The distribution of the data into train and test set could cause a test set not to be representative to the problem, and the resulting error performance can be very high or very low. Taking the average RMSE value of 100 networks reduces the influence of these extreme values.

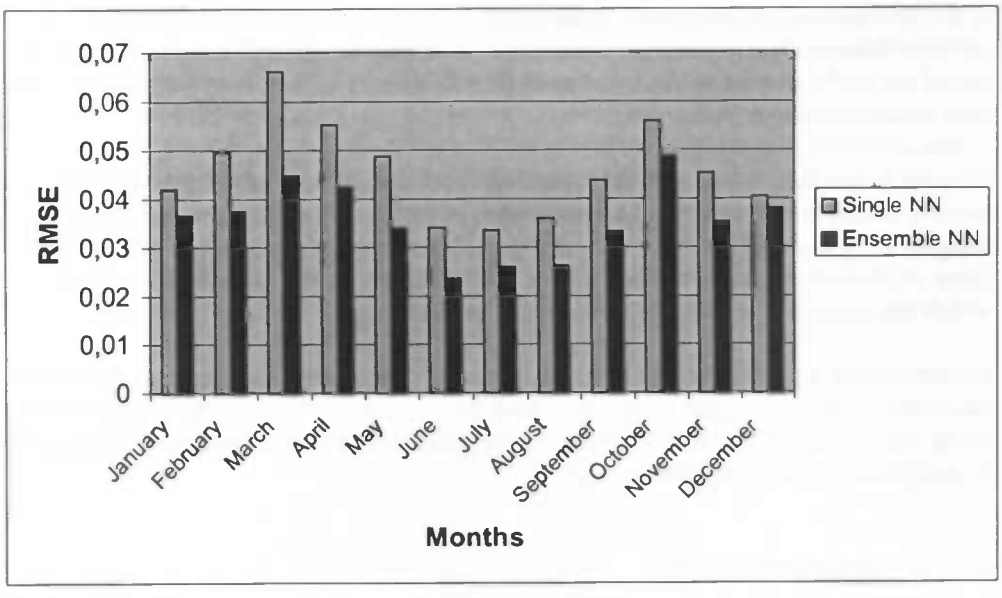

Figure 4-6 Average performance of single neural networks and ensembles trained on the twelve monthly periods of data set A. Each column represents the RMSE value of the networks on test data sets for these periods. As expected the error performance of the ensembles is better.

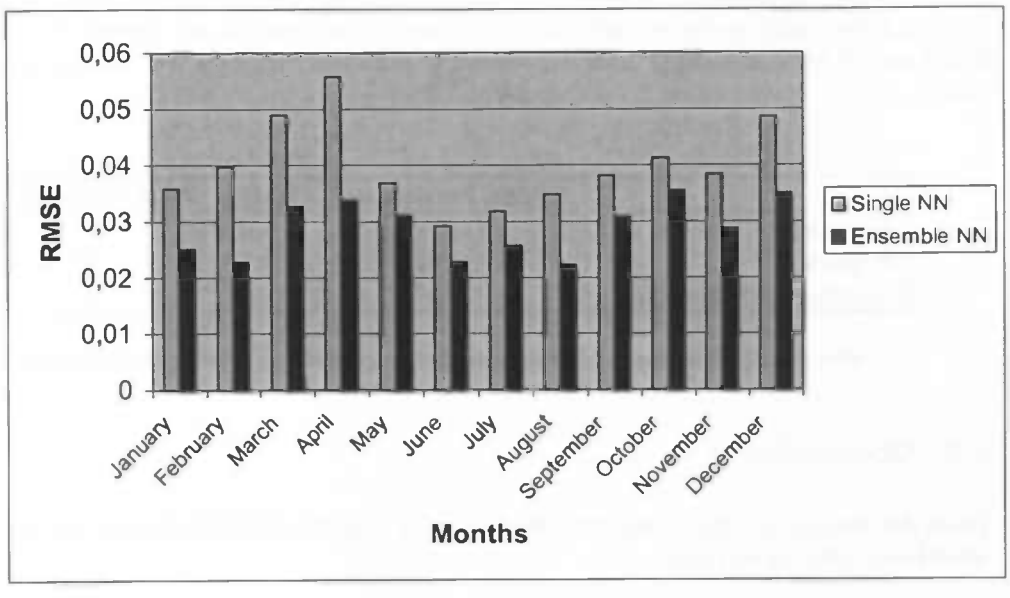

Figure 4-7 Average performance of single neural networks and ensembles trained on the twelve monthly periods of data set B. Each column represents the RMSE value of the networks on test data sets for these periods. As expected the error performance of the ensembles is better.

4.2.4 Mixed ensembles

In the previous section we saw that some neural networks trained on a period of more than one month have a better error performance. This led to the expectation that ensembles of neural networks trained on different periods will achieve higher error-performance than the ones trained on one month.

In order to test this, we altered the balancing combination method. The mixed ensembles contain neural networks trained on bootstraps of different training sets, but the combination weights assignment is based on the error performance of all members of the ensemble on the same validation set, not used for training. This validation set then represents the period for which the ensemble is going to be used as a predictor.

Unfortunately, this did not lead to improvements. The error performance of the mixed ensembles was either equal to or worse than the ensembles of networks trained on bootstraps of the same data set. Because of the lack of available time, we did not have the possibility to investigate the reason of these results.

4.2.5 Persistence model

One of the most simple prediction models is the persistence model, also called the naïve predictor. In this model, the prediction for all times ahead is set to the value now. Our persistence model has a prediction horizon of two days, since the power output measurements are not available earlier. This means that the predicted power output is equal to the measured power output of exactly two days ago. We use this model as a benchmark the final neural network based prediction models have to beat. For the creation of the final models we combined the preprocessing techniques of chapter three and the test results of this chapter. In Table 4-4 the results on the test set of the entire year of both models are shown. The NN based models have an average error improvement of no less than 63% over the persistence model.

	RMSE Persistence	RMSE NN	Decrease
Branch A	0.0916	0.0356	61 %
Branch B	0.0798	0.0277	65 %
Average both branches	0.0857	0.0317	63 %

Table 4-4 RMSE of the persistence model and of the neural network based models

4.3 Conclusion

From the tests in this chapter we draw the following conclusions about the construction of neural networks for this data:

1. Neural networks based on input variables selected by the linear input relevance determination algorithm have an equal or better error performance than networks based on inputs selected with the naïve input relevance determination.
2. Model selection leads to neural networks with better error performance than early stopping.
3. The monthly period is the best period to split the data into.

4. The separation of the data into peak and valley hours, or into week and weekend days worsens the error performance of the networks.
5. Ensembles of 50 networks trained on the same data set result in a better error performance than the average single network and ensembles of networks trained on different data sets.

These models have an average error improvement of 63% over the persistence model.

Chapter 5 Simulating Model Use

The conclusions of the previous chapter holds for the data set used for the training and testing of the neural networks. This data contains weather and power output measurements for the installations of a certain park for the year 2001. The models created in the previous chapter are optimal for the situation of the park in 2001. But when the models will be used in reality they need to predict the power output at that time, and the production circumstances change over time. Not only does the structure of the park change, but the weather will be different as well, and the prediction will be based on weather forecasts instead of measurements.
In this chapter we investigate the influence of this realistic situation by testing the models on data of the same park in 2002. We try to answer the following questions:

1. What is the influence of using weather forecasts compared to using weather measurements?
2. Do the conclusions of the previous chapter based on the 2001 data still hold for 2002?
3. What are the imbalance costs on this data set of the models built on the 2001 data?

5.1 Data

During the last weeks of this research the park data of the year 2002 became available. Throughout that year the park was enlarged from 126 to 128 installations. Unfortunately the data was not complete and contained errors. We corrected most of the missing weather data by copying the weather conditions of previous days, or, when possible, by averaging over the surrounding values. This way we could use data from branch B from the 1st of April 2002 until the 30th of September 2002 for the evaluation of the models built on the 2001 data. The total number of available data patterns was thus 183 days*24 hours*4 quarters = 17568 patterns. The weather information consisted of weather forecasts and the actual weather measurements.

Table 5-1 gives a summarization of the data. Compared to the same branch and period in 2001 the mean sun radiation and temperature have increased a little, and the mean production has slightly decreased.

	Min	Max	Mean	Std	Range	Kurtosis	Skewness
Measured Humidity (%)	20	100	78.63	17.49	80	2.4453	-0.7060
Forecasted Humidity (%)	29	100	80.41	15.30	71	2.8324	-0.7012
Measured Sun Radiation ($J/cm^2/h$)	0	319	59.17	76.60	319	3.3289	1.2126
Forecasted Sun Radiation ($J/cm^2/h$)	0	300	65.84	76.24	300	2.6233	0.9090
Measured Temperature (deg C)	-3	32.60	14.9	5.26	35.6	3.3649	-0.0307
Forecasted Temperature (deg C)	-1.30	30.30	14.60	4.78	31.60	3.3020	0.0142
Measured Wind Direction (deg)	0	360	186.58	97.75	360	2.0164	-0.1277
Forecasted Wind Direction (deg)	0	360	185.01	97.14	360	2.2220	-0.1593
Measured Wind Speed (m/s)	0	13	3.34	1.98	13	3.2065	0.6581
Forecasted Wind Speed (m/s)	0	11	3.68	1.67	11	3.5128	0.6864
Production (P/Pn)	0	0.61	0.17	0.11	0.61	4.0852	1.3178

Table 5-1 Measurements and forecasts of 1 April – 30 September 2002

Figure 5-1 depicts this mean production as the mean daily patterns per three months for 2001 and the available months of 2002. The periods April-June and July-September from both years have very similar patterns. Notice the decrease in mean production at the midday hours of the pattern of April-June, where 2002 has on average a lower power output than 2001.

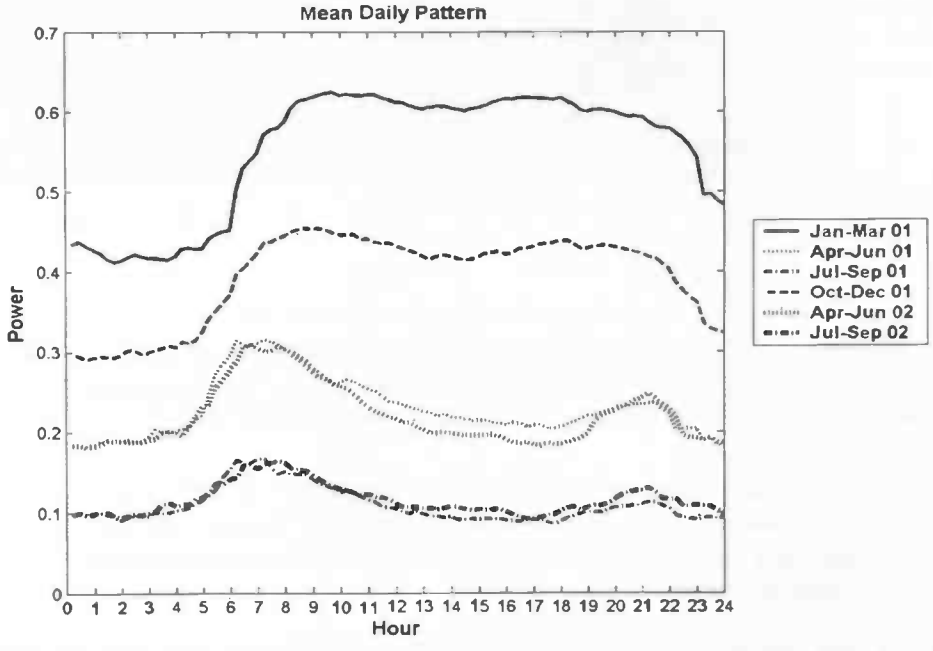

Figure 5-1 Mean daily patterns of the power output of 2001 and the available months of 2002, per three months. Both patterns from 2002 are very close to their counterparts of 2001, although there is a lower power output during the midday hours from April-June 2002.

5.2 Weather forecasts versus weather measurements

In this section we investigate the influence of weather forecasts and weather measurements as input for models built on weather measurements. To test this we compare the performance of the ensembles created on the 2001 data on the 2002 weather forecasts and measurements. The input variables of these ensembles were selected by using the linear input relevance determination. As performance measure we use the imbalance in MWh of the power output prediction.

Figure 5-2 shows the absolute imbalance increase of the networks when run with weather forecasts as input variables relative to the imbalance caused by weather measurements. The numbers next to the bars give the increase in percentage. In April and May the increase is positive, which means that the prediction based on weather forecasts is worse than the prediction based on weather measurements. However, in June, August, and September there is an imbalance decrease, thus the predictions based on forecasts are better in these months.

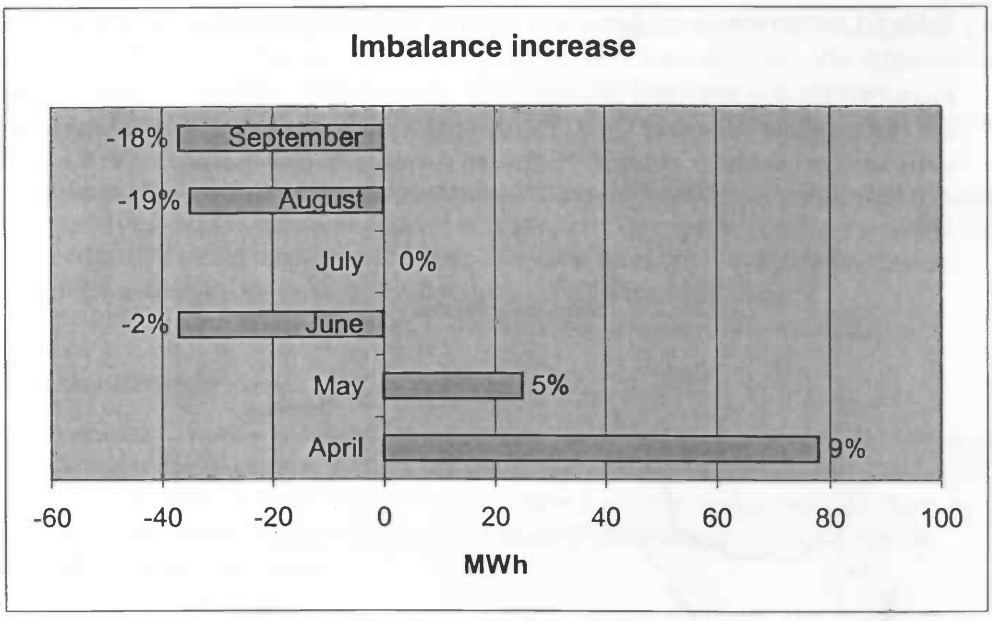

Figure 5-2 Absolute and proportional imbalance increase in MWh of the ensembles when evaluated on weather forecasts instead of weather measurements. In April and May the imbalance of the prediction based on weather forecasts is worse, in June, August, and September the imbalance results are better.

The total difference in imbalance summed over the six months is the surprisingly small decrease of 6 MWh or 0.1%. But the monthly imbalance increases and decreases are not small, especially not the 78 MWh increase in April. So there is a considerable risk in using weather forecasts instead of measurements.

These tests were also performed on other models created on the 2001 data, like single neural networks and networks based on the naïve input relevance variables. The outcome of these tests was similar to the results described above.

Thus, when the prediction is based on weather forecasts instead of weather measurements with networks trained on weather measurements, the changes in imbalance alternate per period, but the total imbalance difference is very small. The absolute imbalance differences per month are not small, and the risk of these large imbalance costs would be important in the analysis of the risks involved in using these models.
We will use weather forecasts for the tests in the remainder of the chapter.

5.3 Evaluating models

Next we will evaluate neural networks that were created on the 2001 data set to test whether the conclusions of section 4.3 still hold. We will use the RMSE value as performance measure.

Input relevance determination

The first conclusion was that neural networks based on input variables selected by the linear input relevance determination algorithm have an equal or better error performance than networks based on inputs selected by the naïve input relevance determination algorithm. To test this we evaluated single neural networks trained on input variables selected by the linear and the naïve input relevance. The networks we used for the tests are the ones with the lowest RMSE value on their 2001 test set. The 2002 data was split up according to the monthly periods used for the creation of the networks.

Figure 5-3 depicts the test results. From April to August the RMSE values of the two input relevance algorithms are almost equal to each other. In September the RMSE value of the model trained on input variables selected by the linear relevance determination is significantly lower.

Thus, the simulation on data from 2002 confirms the conclusion that the neural networks based on input variables selected by the linear input relevance determination have on average an equal or better error performance than networks based on the naïve input relevance determination.

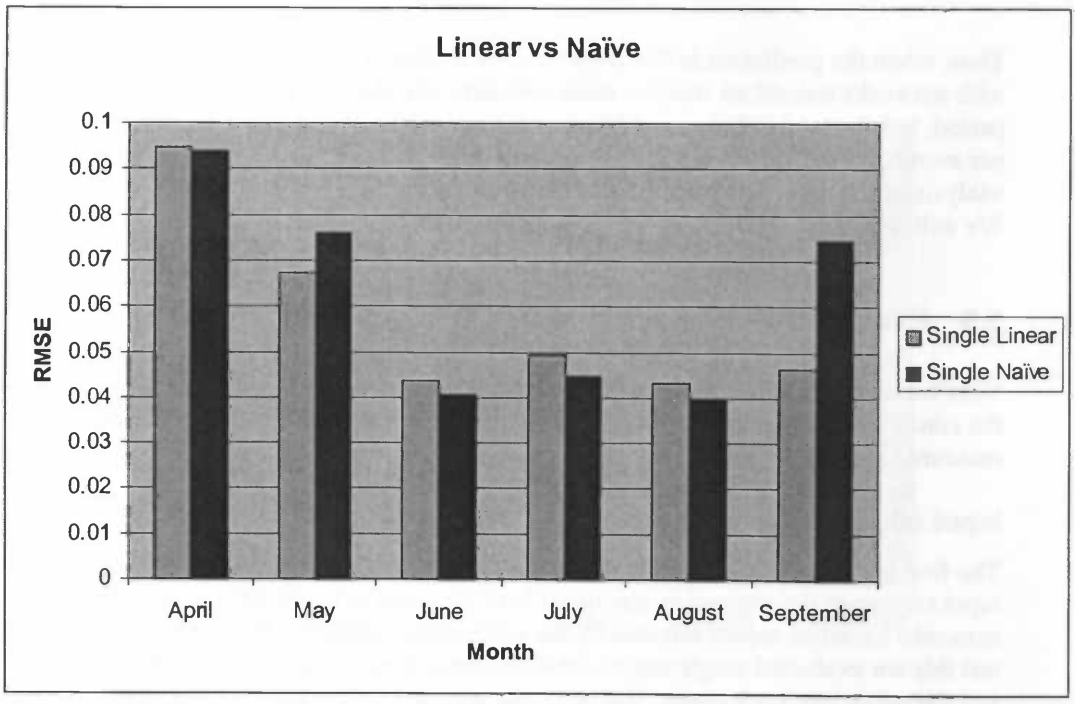

Figure 5-3 RMSE values of the single neural networks with the lowest RMSE value on the 2001 data set, based on input variables selected with the linear and the naïve input selection. The difference in error performance is very small, except for September, where the linear selection results in a significantly lower RMSE value. The networks based on the linear input relevance variables have on average a slightly lower RMSE value.

Because of a lack of time, we did not test if neural networks created with the model selection method would outperform networks created with the early stopping method on the 2002 data. We expect that this would have been the case. The same holds for the conclusion about splitting of training data into peak and valley, or week and weekend days.

Varying the data set size

A more interesting question is whether or not networks trained on data sets of one month still show the best error performance. If there are significant differences per month between 2001 and 2002, then networks trained on larger periods have a smaller chance that they have to extrapolate. Extrapolation generally leads to a bad error performance, since the neural network has to "guess" the output.

To test this, we compared the RMSE on the 2002 data of ensembles trained on periods in 2001 of one month, three months, and one year. The results are depicted in Figure 5-4. While the ensembles trained on a one-month period showed the best error performance on the 2001 data, this is not the case for the 2002 data. In April the yearly model even outperforms the other two, and for the other months the differences are too small to be meaningful.
Based on these results we draw the conclusion that the 2001 data is not representative enough for the underlying problem for monthly periods to have a good generalization performance. One way to solve this is training networks on much larger periods, like a year. Another is to use more representative data when training neural networks, like data from multiple years. The latter solution, however, is only possible if more data is available, and this is not always the case. Furthermore, even multiple years of data do not give the guarantee that you have truly representative data available for training. So the best way to solve this problem is to use larger data periods to build the models on.

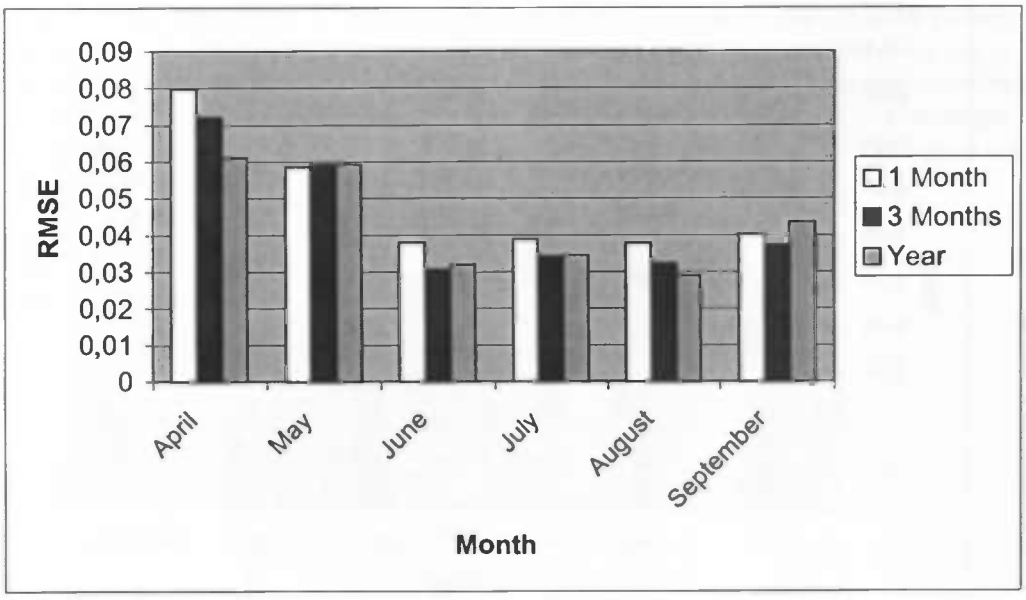

Figure 5-4 RMSE values of ensembles trained on 2001 periods of one or three months or the whole year, evaluated on 2002 data. The RMSE value is mainly equal for the different months, only in April the year model outperforms the others.

The simulation showed that the conclusion that the monthly period is the best period to split the data into does not hold on the 2002 data. Models built on larger periods, like the yearly period, have a better error performance on the 2002 data. Thus, models for the prediction of the power output have to be based on a large period, which is one year for this particular problem.

Ensembles

The last conclusion is that ensembles of networks have a better error performance than the average of single networks trained on the same data set. We tested this by comparing the single neural networks and ensembles that have the best error performance on the 2001 data. While one average the ensembles have a better error performance on the 2001 data, there are for every period single neural networks with a lower RMSE value. These are the single networks we used for this test.

The results of the evaluation are depicted in Figure 5-5. Compared to the best single networks the ensembles have an equal or better error performance on the 2002 data.

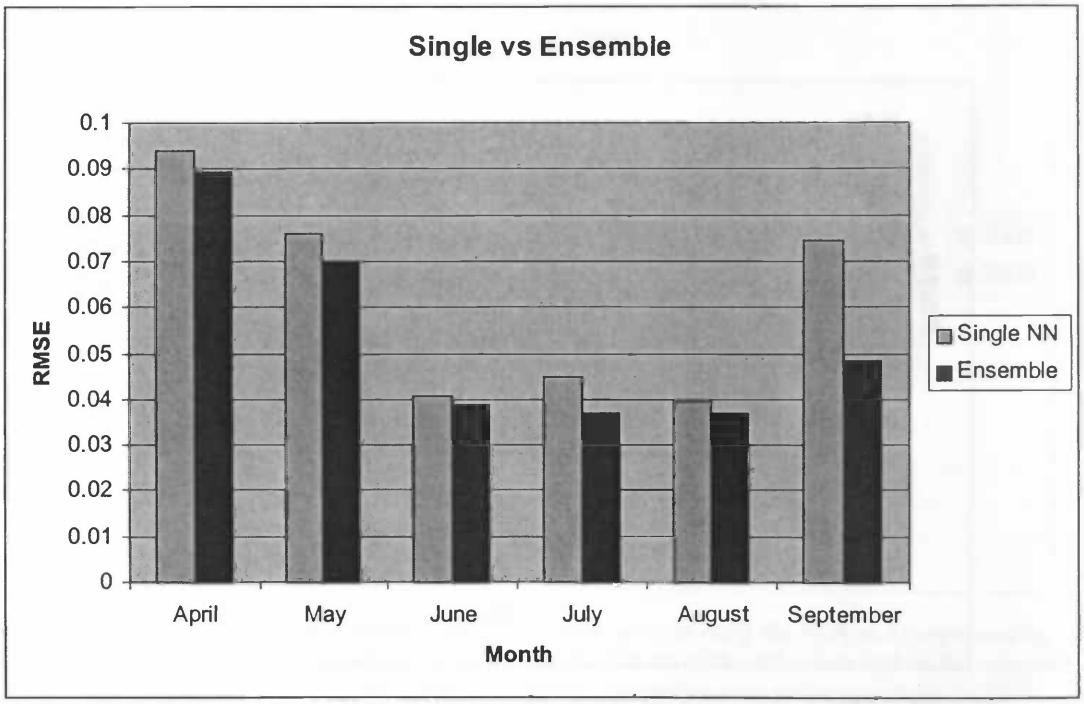

Figure 5-5 RMSE values on the 2002 data of the single networks and ensembles with the lowest RMSE value on the 2001 test set. For all periods the ensembles have an equal or lower RMSE value.

We tested if ensembles of neural networks have a better error performance than the average single neural network, and it holds on the 2002 data. From this we conclude that the ensembles of neural networks have a better generalization performance than the average single neural network. We will therefore keep on using ensembles of neural networks for the prediction models.

Principal Component Analysis

We also evaluated networks trained on data that was not preprocessed using the Principal Component Analysis (PCA). We did this because of the warning that using the PCA does not improve performance when data contains meaningless variables with a high noise level or outliers. Earlier we saw that there exist differences between the two years, which caused the models based on a monthly-period to perform worse than models based on larger a period. These changes could also cause a performance decrease because networks are trained on input data converted by the PCA, according to the optimal conversion of the 2001 data.

The error performance of networks trained and tested on the 2001 data significantly improved by using the PCA. However, tests showed that the error performance on the 2002 data of the same networks decreases due to the use of the PCA. The PCA conversion, based on the 2001 data, probably does not account for the difference in variance and new outliers in the 2002 data, causing some of the new patterns to be wrongly projected. Neural networks then have to base their predictions on wrongly projected input data and maybe even have to extrapolate more than they would have had to otherwise. Hence the performance decreases.

Hence, the differences in the data between the two years cause the model's error performance to decrease when using the PCA as preprocessing technique. We therefore remove the PCA from the preprocessing phase.

5.3.1 Combining the results

During the simulation of model use, we discovered that we needed to change some parts in the construction of the models. The optimal data set size is not one month, but more like a year. Moreover, the use of the PCA for the de-correlation of the input data as preprocessing step does not improve generalization. The removal of this dependency modeling technique alters the preprocessing phase. The new sequence of modeling steps, depicted in Figure 5-6, does not include dependency modeling anymore.

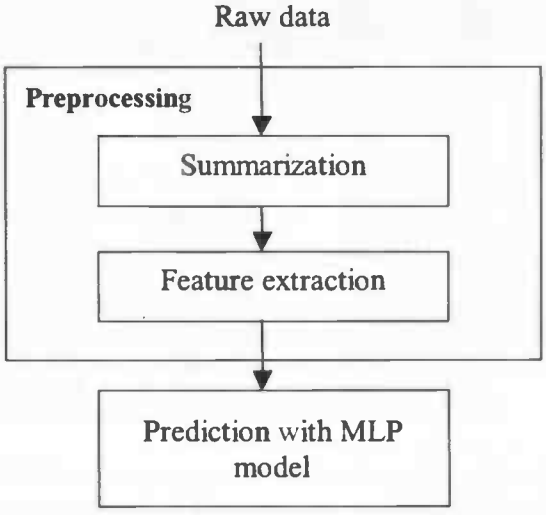

Figure 5-6 Sequence of modeling steps, without dependency modeling as part of the preprocessing phase.

When we combine the results from the simulation tests of the previous section, the construction of the neural networks can be described as follows:

- The most relevant input variables are selected by using the linear input relevance determination.
- The neural networks are trained and tested on data of a yearly period.
- Model selection is used to decide on the optimal number of hidden neurons.
- The models are ensembles of 50 MLP neural networks, combined by the balancing method.

These are the neural networks that we will use to predict the power output. But are these models able to generalize over multiple years? Models built on the 2001 data should then have an equal error performance on the 2001 and 2002 data.

To test this we trained and tested ensembles on a random data split of the complete 2001 data. The RMSE results on both years of these networks are depicted in Figure 5-7. Only for May the RMSE is significantly higher on the 2002 data, the other differences are too small to be considered important. We would have expected a much worse performance on the 2002 data, because earlier tests proved both data sets to be very different. Hence these models have a reasonably well generalization performance.

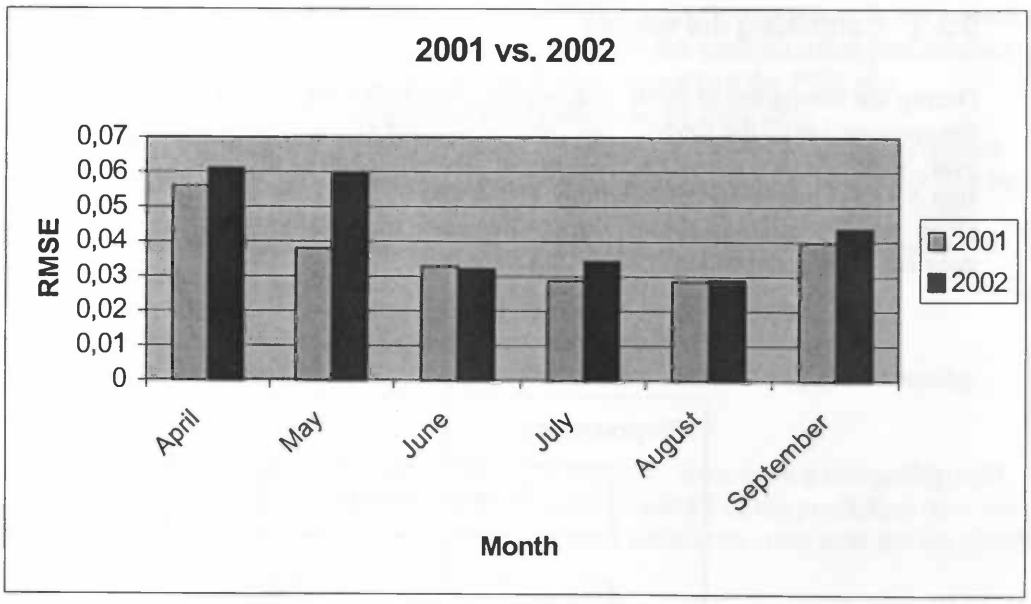

Figure 5-7 Results of ensembles trained on the year 2001, without using the PCA as a preprocessing technique. The RMSE of the models are the results on the test set of the 2001 data and on the complete 2002 data. The only major difference in RMSE is in May, where the error performance on the 2001 data is better.

Still, the generalization performance and thus the error on the 2002 data will probably improve when this data is used next to the 2001 data for the creation of the models. The models can then be based on two years of information about the park. This is highly likely to be a better representation of the park in later years than data only from 2001.

5.3.2 Imbalance costs

Another method for comparing the performance of the neural networks is to calculate the imbalance costs of the prediction error. As a simplification we used 4 Euro per wrongly predicted MWh.

We evaluated models on the available data from 2002 that are trained and tested on the 2001 data. We compared five types of neural networks discussed earlier. They are:

A. Single networks based on monthly data and input variables selected by the linear input relevance determination.
B. Single networks based on monthly data and input variables selected by the naïve input relevance determination.
C. Ensembles based on monthly data and input variables selected by the linear input relevance determination.
D. Ensembles based on monthly data and input variables selected by the naïve input relevance determination.
E. One ensemble based on the whole year and input variables selected by the linear input relevance determination, and without using the PCA as a preprocessing technique (the model described in the previous section).

The results are shown in Figure 5-8. The ensemble E produced with a total of €6088 over the six months the lowest imbalance costs. Furthermore, the differences between models based on the linear and on the naïve input relevance determination are more evident than when we

compared them by their RMSE. This results from the different types of performance measures: the RMSE counts large differences between the prediction and the actual power output less heavy. The prediction error caused by the linear input relevance determination algorithm has less extremely large values than the prediction error caused by the naïve algorithm, and therefor the difference between the linear input relevance algorithm and the naïve input relevance algorithm is larger when compared by imbalance costs.

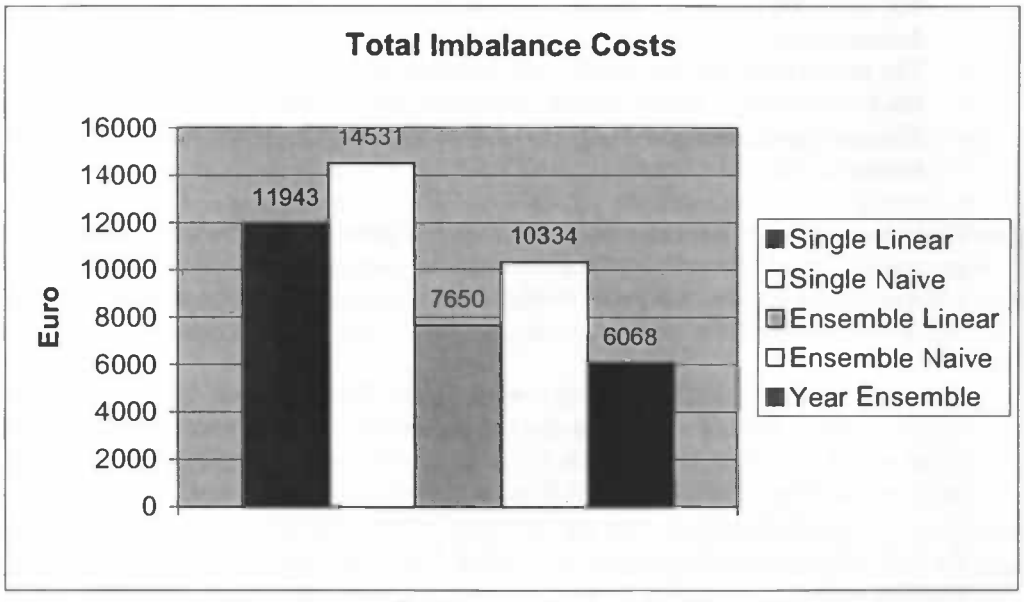

Figure 5-8 Imbalance costs summed over six months of five types of networks trained on 2001 data: single networks and ensembles of networks, based on inputs selected by the linear and the naïve input relevance determination, and a yearly ensemble based on input variables selected by the linear relevance determination without PCA as preprocessing technique. The last one has the lowest imbalance costs summed over 6 months of 2002.

5.4 Summary

In this chapter we simulated model use by investigating the influence of new data on the performance of the models created in the previous chapter. The new data consists of weather forecasts and weather and power output measurements, belonging to the same park, but from a period of six months of 2002, one year later.

We discovered that on average there is not a big difference when weather forecasts are used as model input instead of weather measurements, but that the imbalance difference in MWh per period can be quite large, so the risks involved are not small.

We tested whether or not the conclusions made in chapter 4 about the construction of neural networks are also valid for the 2002 data. The first conclusion remains valid: the use of the inputs selected by the linear relevance determination results in equal or better performing models on the simulation data than the use of the inputs selected by the naïve relevance determination. This small difference is more pronounced when we look at the imbalance costs of the prediction instead of the RMSE. Furthermore, ensembles of neural networks remain to have an increase in generalization performance when compared to single networks.

But the assumption of one month as the optimal period to base the models on is not correct. Models built on large data periods, like one year, have a better generalization performance than models built on data of monthly periods. We also learned that the PCA as a preprocessing technique does not improve performance on the 2002 data.
We altered the construction of the models in accordance with the results of the simulation. Figure 5-6 shows the new sequence of modeling steps, where the dependency modeling by the application of the PCA is left out. The new construction of the models is as follows:

- The most relevant input variables are selected by using the linear input relevance determination.
- The neural networks are trained and tested on data of a yearly period.
- Model selection is used to decide on the optimal number of hidden neurons.
- The models are ensembles of 50 MLP neural networks, combined by the balancing method.

These models, when trained and tested on the complete 2001 data, have a reasonably well generalization performance on the 2002 data: the average RMSE of the prediction does not differ a lot between the two years. Furthermore, the total imbalance costs over the six months of 2002 is the lowest for these networks.

The improvement in RMSE of these models on the 2002 data over the persistence model would be about 49%, if we assume that the persistence model's error performance remains equal on the 2002 data, compared to the error on previous year's data. This is 16% lower than the improvement of 65% on branch B we achieved on the 2001 data (see section 4.2.5). So there still is a difference in error performance between the two years, and we expect that when the 2002 data would also be used for training, the performance of these models would increase.

Chapter 6 Discussion

In the previous chapters, it has been shown how the prediction of the power output of non demand-driven power systems can be modeled by using neural networks. Data analysis methods were described and applied to the problem for preprocessing. Models based on neural networks were created on data from one year, the modeling data. The real-life situation was simulated by evaluating the models on data from part of the next year, the simulation data. The conclusions of the previous chapters are briefly summarized in this chapter. Finally, a few ideas for further research are given.

6.1 Conclusions

This work shows that neural networks, specifically the MLP models developed in chapter four and five, are suited for the modeling and prediction of the output of non demand-driven power systems. This was achieved by combining the static MLP with preprocessing techniques to extract the time components from the input data.
The major challenge of this research was to create models that can accurately predict the power output in order to minimize the imbalance costs. The problem, however, always is that the models have to be based on a data set that not necessarily is representative for the underlying problem and the real-life application of the models. This was confirmed when not all of the conclusions we drew in chapter 4 about the building of the models held when the model use was simulated in chapter 5 on the simulation data. We found that for better generalization performance the optimal data set size for building models is not 1 month, but more like a year, and that the application of the Principal Component Analysis as preprocessing technique does not improve generalization. The construction of the models was altered according to these results.
Furthermore, ensembles of neural networks combined via the balancing algorithm have a better generalization and error performance than single networks.

Figure 5-6 shows the final sequence of modeling steps. The preprocessing consists of summarization and feature extraction, as described in Chapter 3. MLP based models are used for the prediction, the construction of these models is as follows:

- The most relevant input variables are selected by using the linear input relevance determination.
- The models are trained and tested on a yearly period.
- Model selection is used to decide on the optimal number of hidden neurons.
- The models are ensembles of 50 MLP neural networks, combined by the balancing method.

Table 6-1 shows the performance of the model built on the modeling data. The first two columns hold the average RMSE value of the model on the period April to September from the two consecutive years. This is because the simulation data was available only for this

period. The RMSE on both data sets is not very different, while the model is built on just the modeling data. This means that the model has a reasonably well generalization performance. The improvement over persistence of both data sets is the average RMSE improvement of the model over the persistence model on the same data. The estimated imbalance costs of the prediction on the available months of the simulation data are given in the last column.

RMSE modeling data	RMSE simulation data	Improvement over persistence modeling data	Improvement over persistence simulation data	Imbalance costs (Euro) on simulation data
0.037	0.041	54%	49 %	6068

Table 6-1 Performance of the final model. The RMSE values are the mean values over the period April to September. The improvements are the percent improvements over the RMSE of the persistence model on the same data. The last column holds the imbalance costs of the final model in Euro.

So the final models we created have a reasonably well generalization performance: the RMSE did not get much higher on the simulation data, which the network had never seen before. Furthermore, this network has the lowest imbalance costs on this data. When compared to the persistence model, the final models showed a better error performance of 49%.

A big limitation of real life problems is the dependence of models on real-life data. The available data was incorrect and may not be representative for the problem. This is a general problem and more than trying to correct the data as well as possible and avoiding overfitting of the data-specific and problem unspecific elements one cannot do.

Another problem with models based on neural networks lies in the reliability; the predictions are created in a rather complicated way and it is very difficult to understand how the model works. Therefore, the behavior in abnormal conditions may be unexpected. For this reason, a thorough on-line testing is necessary in order to ensure the reliability in varying conditions. Only this will provide a definite opinion on the applicability of the model.

6.2 Ideas for further research

One future refinement to the models would be to include the simulation data in the training process. We expect that this would increase the error performance of the models by another 5-10%.

The principal component analysis we applied and discarded in this research is a linear, unsupervised technique for de-correlation. Many alternatives exist, among which are nonlinear and supervised de-correlation techniques. Using one of the alternatives to preprocess the input data could result in a better prediction by the models.

We used the multilayer perceptron to model the prediction and applied preprocessing techniques to capture the time factors of the input data. Another possible approach would be to use time-delay neural networks, and to let the network deal with the time factors. It is possible that the problem underlying the power output data is subject to certain time factors, which the methods we used are not able to capture. Recurrent neural networks that adapt their feedback loops during training might be able to find these time factors, and have a better performance in the prediction of the power output.

A very interesting research would be to use adaptive neural networks for the prediction. These networks continue to adapt themselves to the data they are fed when in use. The neural network can then learn changes in the power output, caused by changes in the structure of the park, without needing to go through the training and implementing process all over again. Furthermore, the field of adaptive neural networks is still relatively new and many things are unknown.

Appendix A

Modeling

The interpretation of data by means of modeling is an important method to solve problems. But although models are very powerful in describing situations and looking at the future, they will never be able to completely represent reality. In most cases the complexity of the problem, lack of knowledge about the situation, and errors in the data cause a model to only be able to give an approximation.

A.1 The Modeling Process

According to [1] the simplification of the reality in modeling depends on a set of essential factors:

- The subject and the goal the model is used for
- The subjective view of reality of the model builder
- Certain external limiting conditions

Another characteristic is the iterative way a model is developed. The modeling process is a more or less sequential set of steps to be taken, and during each step it might be necessary to make one or more steps backwards. The modeling process in this section refers to the modeling process as defined by Janssen, Slob and Rotmans [11]. The complete iterative modeling process is shown in Figure A-1.

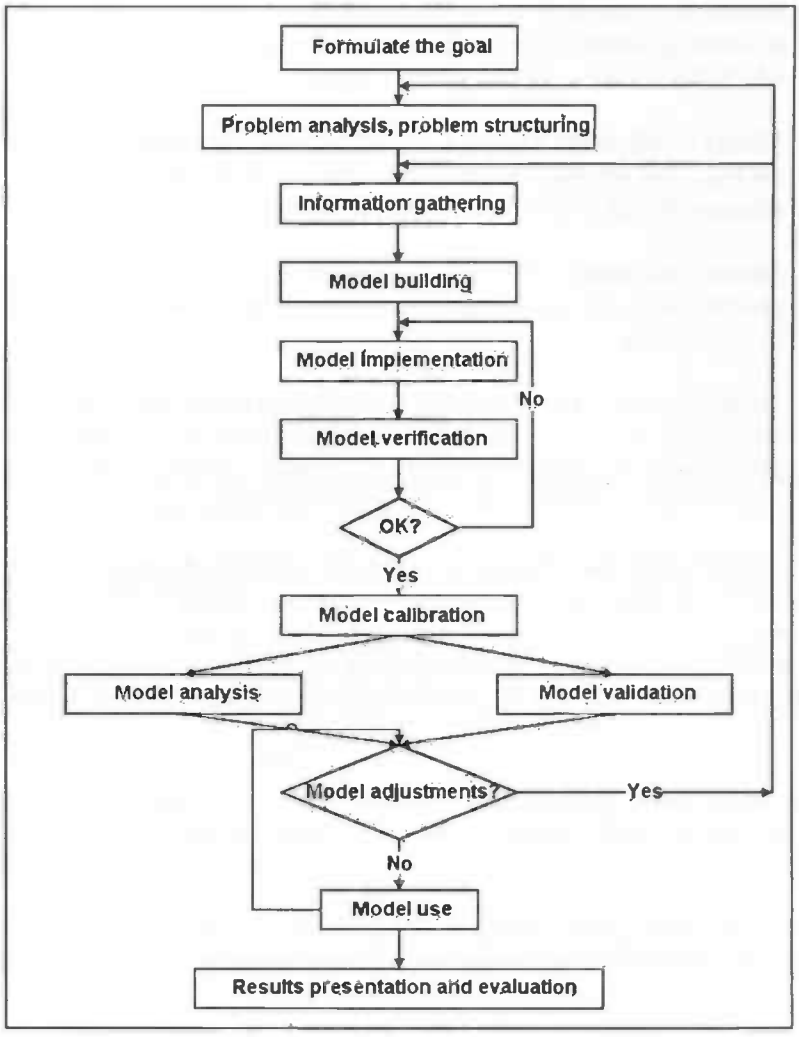

Figure A-1 The modeling process after [11]

Below a description is given of the steps of Figure A-1.

1. **Formulate the goal**: The model goal is the first item that should be defined. Examples are control, diagnosis, and prediction.

2. **Problem analysis and problem structuring**: The problem analysis and structuring involve partitioning the goal into relevant sub-goals and their relations, describing the boundaries of the system, and determining what kind of data plays an important role. It is very important to make no mistakes in this part of the modeling, because they are hard to recover from later on.

3. **Information gathering**: Collecting the data needed in next steps of the modeling process. This may also involve data preprocessing and data analysis.

4. **Model building**: This crucial step involves the building of the model or sub-models. Choices have to be made about necessary simplifications and about which part of reality the model should represent.

5. **Model implementation**: The model implementation involves some practical choices concerning hardware and software, and the implementations of the model relations, which method of approach should be taken, etc.

6. **Model verification**: During and after the implementation the program has to be tested for bugs and wrongly implemented relations. If the program is declared to be adequately free of errors, the next step can be taken.

7. **Model calibration**: The unknown quantities of the model, like parameters and preconditions, are initialized in such a way that the model's results best reflect the available data.

8. **Model analysis**: The purpose of analysis is to know more about the characteristics of the model, relevant factors, which small variations in a variable cause a large difference in the simulation result, and model confidence. Better and more detailed data and/or sub-models can result from this phase.

9. **Model validation**: During this stage the model has to prove its validity, for example by its capability to produce simulation results similar to historical empirical data.

Model building and model implementation are not always as separable as described in the modeling process by [11]. For this kind of research the two steps are considered as one.

References

[1] Benders, R.M.J., "*Interactive simulation of electricity demand and production*", Groningen: R.M.J. Benders, 1996.

[2] Breiman, L., "Bagging Predictors", *Machine Learning*, 24:123-140, 1996.

[3] Hansen, L.K. and Salamon P., "Neural network ensembles", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 12 (10), pp 993-1001, 1990.

[4] Haykin, S., "*Neural Networks, a comprehensive foundation, second edition*", New Jersey: Prentice Hall International, Inc., 1999.

[5] Helliwell, I. S., Turega, M. A., Cottis, R. A., "Accountability of neural networks trained with 'real world' data". *Artificial Neural Networks, Conference Publication*, Vol. No. 409 IEE. pp 218-222, 26-28 June 1995.

[6] Heskes, T., "Balancing between bagging and bumping", In M. Mozer, M. Jordan, T. Petsche, editors, *Advances in Neural Information Processing*, pp 466-472, Morgan Kaufmann, 1997.

[7] Heskes, T., "Selecting weighting factors in logarithmic opinion pools", In M.I. Jordan, M.J. Kearns, S.A. Solla, editors, *Advances in Neural Information Processing*, Vol. 10, pp 466-472, the MIT Press, 1998.

[8] Holden, Sean B., Rayner, Peter J. W., "Generalization and PAC Learning: Some New Results for the Class of Generalized Single-Layer Networks", *IEEE Transactions on Neural Networks*, Vol.6, No. 2., March 1995.

[9] Hornik, K., "Approximation capabilities of feedforward neural networks", *Neural Networks*, Vol. 4 pp 251-257, 1991.

[10] Jacobs, R.A., "Methods for combining experts probability assessments", *Neural Computation*, Vol. 7(5), pp. 867-888, 1995.

[11] Jansen, P.H.M., Slob, W., Rotmans, J., *"Gevoeligheidsanalyse en Onzekerheidsanalyse: een Inventarisatie van Ideeën, Methoden en Technieken*", National Institute of Public Health and Environmental Protection, Bilthoven, The Netherlands, 1990.

[12] Jongepier, A.G., "*Artificial Neural Networks Applied to Power Systems*", Wageningen: Ponsen & Looijen bv, 1996.

[13] Van de Laar, P., Heskes, T., Gielen, S., "A New Approach to Input Relevance Determination", Submitted, 1997.

[14] Op 't Landt, F.W., *"Stock Price Prediction using Neural Network"*, Master's thesis, Leiden University, 1997.

[15] Manly, Bryan F.J., "*Multivariate Statistical Methods, A primer*", London: Chapman & Hall, 1994.

[16] McCulloch, W., Pitts, W., "A logical calculus of ideas immanent in nervous activity", *Bulletin of Mathematical Biophysics*, 5:115-133, 1943.

[17] Murto, Pauli, "*Neural Network Models for Short-Term Load Forecasting*", Master's Thesis, Helsinki University of Technology, 1998.

[18] Naftaly, U., Intrator, N., Horn, D., "Optimal ensemble averaging of neural networks", *Network*, vol. 8, pp. 283-296, 1997.

[19] Rumelhart, D., Hinton, G., Williams, R., "*Parallel Distributed Processing: Explorations in the Microstructures of Cognition",* Cambridge, MA: MIT Press, 1986.

[20] Reed, Russell D., Marks, Robert J., "*Neural Smithing*", Cambridge: The MIT Press, 1999.

[21] Sandberg, I.W., Xu, L., "Uniform approximation of multidimensional myopic maps", *IEEE Transactions on Circuits and Systems*, vol. 44, pp. 477-485, 1997.

[22] Sandberg, I.W., Xu, L., "Uniform approximation and gamma networks", *Neural Networks*, vol. 12, pp. 781-784, 1997.

[23] Sarle, W.S., ed. (1997), *Neural Network FAQ, part 1 of 7: Introduction,* periodic posting to the Usenet newsgroup comp.ai.neural-nets, URL: ftp://ftp.sas.com/pub/neural/FAQ.html.

[24] Stark, J., "*Nonlinear Dynamics II, III: Analysis of Time Series*", Centre for Nonlinear Dynamics and its Applications, University College London, London, URL: http://www.ucl.ac.uk/CNDA/courses/coursework/Model_Uncert.pdf.

[25] Tibshirani, R., Knight, K., "*Model search and inference by bootstrap "bumping"*", Technical report, University of Toronto, 1995.

[26] Wang, Xue Z., "*Data Mining and Knowledge Discovery for Process Monitoring and Control*", Springer, 1999.

[27] Van Wezel, M.C., *"Neural Networks for Intelligent Data Analysis - Theoretical and experimental aspects*", University Leiden, 2002.